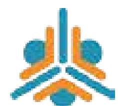
WORK PACKAGE (WP) 1

"TESTING SOFTWARE PROGRAMS"

"SOFTWARE DEVELOPMENT"



Project "Youth networking for economic exchange in the cross-border region"
Project No. CB007.2.22.078

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

CONTENTS

WORK PACK (WP) 1 1

THEMATIC CYCLE 1: "TESTING SOFTWARE PROGRAMS" 4

1. BASICS OF TESTING: OBJECTIVES, TESTING AND DEBUGGING, QUALITY ASSESSMENT, DEFECTS, ROOT CAUSES AND EFFECTS; 4
2. PRINCIPLES OF TESTING: 4
3. TEST OPERATIONS AND TASKS: TEST PLANNING, TEST MONITORING AND CONTROL, TEST ANALYSIS, TEST DESIGN, TEST EXECUTION, TEST EXECUTION, TEST COMPLETION; 5
4. TEST STRUCTURE: DESIGNING AND PRIORITIZING TEST CASES AND TEST CASE SETS, IDENTIFYING THE NECESSARY TEST DATA IN SUPPORT OF TEST CONDITIONS AND TEST CASES, DESIGNING THE TEST ENVIRONMENT AND IDENTIFYING THE NECESSARY INFRASTRUCTURE AND TOOLS, CAPTURING TWO-WAY TRACEABILITY BETWEEN THE TEST BASE, ETC. 6
5. PRODUCTS FOR TEST WORK: TRACING BETWEEN THE TEST BASE AND THE TEST PRODUCTS; 7
6. SOFTWARE DEVELOPMENT LIFE CYCLE MODELS: 7
7. TEST LEVELS: COMPONENT TESTING, INTEGRATION TESTING, SYSTEM TESTING, ACCEPTANCE TESTING; 11
8. TEST TYPES: FUNCTIONAL TESTING, NON-FUNCTIONAL TESTING, TESTING IN A WHITE BOX; 11
9. RISKS AND TESTING; 13

THEMATIC CYCLE 2: "SOFTWARE DEVELOPMENT" 15

L. BASICS OF PROGRAMMING: 15

WHAT DOES "PROGRAMMING" MEAN? 15

TYPES OF CONDITIONAL STRUCTURES 19

FOR LOOP 24

2. ALGORITHMS, ELEMENTS OF C/C ++ PROGRAMMING LANGUAGES, BASIC DATA TYPES; 27
3. CONSISTENT AND CONDITIONAL EXECUTION, ITERATIVE SOLUTIONS, FUNCTIONS; 31

4 ARRAYS, MATRICES AND THEIR APPLICATIONS; 35

5. STRINGS PROCESSING ELEMENTS; 39
6. STRUCTURES 41
7. OBJECT ORIENTED PROGRAMMING (OOP) 51

CLASS 53

OBJECTS 53

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

CLASS ... 53
OBJECTS ... 53
8. DEVELOPMENT OF LARGE SYSTEMS ... 53
9. DATABASES FOR SYSTEM DEVELOPERS ... 55
DATABASE MANAGEMENT SYSTEM ... 58
MAIN CHARACTERISTICS OF DBMS: ... 58
DATABASE AIDS ... 59
INQUIRY LANGUAGES ... 59
DATA DICTIONARIES ... 59
AIDS FOR MONITORING AND EVALUATION ... 59
REPORT GENERATORS ... 59
TYPES OF DATABASES ... 59
NETWORK DATABASE MODEL ... 59
RELATIONAL DATABASE MODEL ... 59
10. DEVELOPMENT OF CONTRACT-BASED SYSTEMS ... 60
11. SYSTEM INTEGRATION ... 61
PRACTICAL EXERCISES AND TESTS ... 68
THEMATIC CYCLE 1 "SOFTWARE TESTING" ... 68
THEMATIC CYCLE 2: "SOFTWARE DEVELOPMENT" ... 74
THEMATIC CYCLE 2: "SOFTWARE DEVELOPMENT" ... 94
THEMATIC CYCLE 2: "SOFTWARE DEVELOPMENT" ... 104

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

Thematic cycle 1: “Testing software programs”

1. Basics of testing: objectives, testing and debugging, quality assessment, defects, root causes and effects;

Testing is the process of examining software or parts of software in a controlled environment and under controlled circumstances in order to detect deviations from requirements (requirement specifications) and verify that the system meets the defined acceptance criteria.

Software testing aims to produce quality, defect-free software that works as expected. Testing is a process that evaluates the functionality of a system to determine whether it meets predefined requirements or not. Identified defects are corrected and tested until it can be ensured that the product is defect-free. Both the overall operation of the system and individual functionalities can be tested.

During the testing, errors, omissions or missing requirements in contradiction with the set requirements are identified. It can be done manually or with the help of automated tools. Some prefer software testing such as White Box and Black Box Testing.

2. Principles of testing:

1) Testing shows the presence of defects: Testing may show the presence of defects/bugs, but cannot be evidence that there are no defects. Even after testing the product, we cannot say 100% that there are no defects. Testing always reduces the number of undetected bugs left in the software, but even if there are no defects, it is not proof of correctness.

2) Full testing is not possible: It is impossible to test everything, including combinations of introductions and prerequisites. Instead of doing all of this, we can use risks and priorities to focus on testing attempts. For example, an application has 15 input fields, each with 5 possible values, to test each possible combination we will need 30,517,578,125 tests (5 to the 15th degree). It is unlikely that the company will give you enough time to run these tests. That is why setting risks and priorities are some of the most important things in every project.

3) Early testing: In the software developing life cycle (SDLC), testing should start at the earliest possible stage and should focus on defined tasks.

4) Defect clustering or bug accumulation: A small number of modules contain the largest percentage of defects found in pre-release testing.

5) Pesticide paradox: If any tests are repeated again and again, eventually the same tests will show the same thing and stop finding other new bugs. To overcome the Pesticide paradox, we

need to review test cases frequently and do new and different tests to find other undetected bugs.

6) Testing depends on the context: Testing is generally context dependent. Different things are tested for different bugs and critical software (the one that is related to human lives) is tested for different things from ordinary software and has to cover some norms

7) The delusion of lack of defects: If the system itself/the application itself cannot be used by the users for what it is intended for or does not fulfill the given conditions and requests of the users, then finding and fixing defects does not help.

3. Test operations and tasks: test planning, test monitoring and control, test analysis, test design, test execution, test execution, test completion;

The visible part of testing is the execution of the tests. However, in order for the tests to be effective and efficient, the need for time to plan tests, construct test cases, prepare for tests and assess the condition of the software based on the test results should be taken into account.

The testing process includes the following **main operations:**

- Test planning
- Control
- Analysis and design
- Execution
- Setting criteria for completing the test and reporting the results
- Operations related to test completion

In some tests, these operations are defined by the term basic software testing processes. Although logically linked, these basic operations can be performed simultaneously, not necessarily sequentially

Testing plan

The testing plan materializes the testing strategy, describes the resources that will be used and the environment, the environment in which the testing will be executed, the constraints that will be applied and the schedule for performing the tests. Testing plans are different for different levels.

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

4. Test structure: designing and prioritizing test cases and test case sets, identifying the necessary test data in support of test conditions and test cases, designing the test environment and identifying the necessary infrastructure and tools, capturing two-way traceability between the test base, etc.

A test case is a series of actions that are performed to determine a specific function or functionality of your application.

Identifying test cases can take a long time, and sometimes you need to repeat the test. Therefore, it must be documented. The following **elements** must be recorded for each test:

- The test case should have the expected result.
- The test case may have preconditions, such as certain values in the database, which must be present in advance.
- The test case may also include post conditions that apply after the test case is completed.
- During the test case, you document the results that were observed in the actual results column.

Design of test cases

The login test format contains the following format:

- Test case ID
- Part of the test script
- A test is being performed
- Test data
- Expected results
- Actual results
- Test result (successful or not)

Best practices for a good test case

- Test cases should be simple and transparent
- Perform a test with the end user
- Do not take a working application while preparing the test case. Stay with the requirements and design documents.
- The test case testing must meet the requirements
- The test case should generate the same results each time, regardless of who runs the test.

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

5. Products for test work: tracing between the test base and the test products;

The test conditions must be able to be related to their sources in the test base, this is known as traceability. Traceability can be horizontal throughout the test documentation for a given test level (e.g. system testing, from test conditions through test cases to test scripts) or it can be vertical across the development documentation layers (e.g. from component requirements).

Traceability refers to a document in software testing that intends to establish a link between various factors. It helps to determine the completeness of a relationship by comparing basic documents (business and technical). The traceability is checked by preparing a tabular presentation of data (test cases). Tracking can range from mapping requirements to components, test scripting conditions, or test cases.

The test analysis work products include defined and prioritized test conditions, each of which is ideally bidirectional to track the specific test element(s) based on the test it covers. Test analysis can also lead to the detection and reporting of defects in the test base.

Advantages of traceability:

- We get a clear picture of whether the software project is developed according to the set requirements.
- To make sure that all requirements are included in the test cases.
- To avoid adding unnecessary or inappropriate features to the developed project.
- Missing functionalities can be easily identified using different types of tracking.
- Updating test cases can be done easily if there are changes in the requirements.
- Confirms 100% test coverage.

6. Software development life cycle models:

Knowledge of the software development process provides a basis for understanding the causes of software errors and how they are implemented in applications.

6.1. The **life cycle** of a software system usually includes the following phases:

- Requirements analysis

The software life cycle begins with compiling a set of requirements for it. The requirements analysis is performed to assess the technical feasibility, to determine how to divide the software system, to identify the specifications that need to be agreed, to establish for each individual part to which the system refers. The result of such an analysis is a specification of requirements, also called a development assignment.

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

- Design

The design phase involves the construction of the individual parts of the software system. Activities in this phase include designing the technical architecture of the software, designing the database, designing user interfaces, selecting or creating new algorithms, i.e. creating a technical project.

- Programming

The activities in this phase include the creation of test data, the creation of program code (Source Code), object code, working documentation, integration plan and its implementation.

- Testing

The testing phase aims to detect and eliminate errors in the system. Defects found at this stage are sent for correction and retested. This process is repeated as many times as necessary to meet the requirements for the degree of bugs that remain unfixed.

- Installation and experimental operation

Once the software is tested, it is provided to the customer. The test plan usually contains a schedule and norms for performing tests in the experimental operation mode and a method to collect, report and eliminate bugs after commissioning. Operations in this phase are: installation planning, software delivery, user training, software installation.

- **Operation and maintenance** Regular operation performs ancillary activities. This includes the provision of technical assistance, the maintenance of a regime for responding to requests for assistance from the applicant

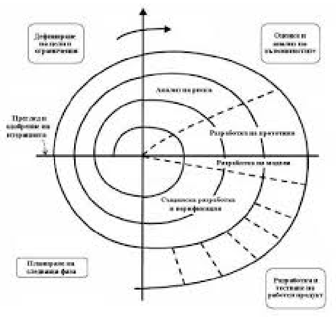
.

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

6.2. Life cycle models

As software development processes are linear, there are several **theoretical models** that reflect the different forms of software life cycle flow. Each of these models has its advantages and disadvantages.

Model	**Strengths**	**Weaknesses**
Waterfall	ü Each phase must be completed in order to proceed to the next ü Early planning is needed ü Testing is an integral part of the product life cycle ü Quality completion of each phase	- Depends on the definitions and requirements that are established from the earliest stages of the cycle. - Depends on the separation of design requirements - The feedback is only from the testing phase - It is focused on the whole product, not on the individual processes
Prototyping	ü Requirements can be defined earlier and more reliably ü The requirements can be researched quickly and at a low cost ü Errors in requirements and design are detected at an early stage	- Requires prototyping tools and experience with them, which is a development expense - The prototype can become a production system

Spiral 	ü Reuse of existing software ü Eliminates errors ü Balances resource costs ü Quality objectives are formulated in the development process ü The product may evolve ü Provides a reliable framework with an integrated hardware and software development system	- It is related to Rapid Application Development, which is very difficult in practice - The process is difficult to manage

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

7. Test levels: component testing, integration testing, system testing, acceptance testing;

The most commonly used testing methods are component testing, integration testing, acceptance testing, and system testing. The software goes through these tests in a certain order.

7.1. Component testing

First, a single test is performed. As the name suggests, this is an object-level testing method. Individual software components are tested for errors. This test requires accurate knowledge of the program and each installed module. In this way, this verification is performed by programmers, not testers. For this purpose, test codes are created, which check whether the software behaves as intended.

7.2. Integration testing

Individual modules that have already been tested on units are integrated with each other and checked for faults. This type of testing mainly detects errors in the interface. Integration testing can be performed using a top-down approach, following the architectural design of the system. Another approach is the bottom-up approach, which is carried out from the bottom of the control flow.

7.3. System testing

In this test, the entire system is checked for errors. This test is performed by connecting hardware and software components of the entire system, after which it is checked. This testing is listed under the "black box" testing method, where the expected user conditions of the software are checked.

7.4. Acceptance tests

This is the last test that is performed before handing over the software to the customer. It is conducted to ensure that the software that is developed meets all customer requirements. There are two types of acceptance tests - one conducted by members of the development team, known as internal acceptance testing (Alpha testing), and the other, which is conducted by the customer, known as external acceptance testing. If testing is done with the help of prospective customers, it is called customer acceptance testing. If testing is done by the end user of the software, it is known as acceptance testing (beta testing).

8. Test types: functional testing, non-functional testing, testing in a white box;

8.1. Functional tests

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

Functional testing checks the behavior of the system by entering input and output data. The "Black box" testing method is used. The tests are created based on the functional requirements. Functional requirements determine the behavior of the system. They define its limitations. The functional requirements must be documented in: - requirements management system; - software requirements specifications (SRS). The requirements are used as a basis for testing. At least one test is created for each requirement. More than one test is required to meet the requirements. Requirement-based testing is mainly used in system testing and acceptance testing.

8.2. Non-functional tests

Application security is one of the main tasks of the developer. The security test checks the software for confidentiality, integrity, authentication, availability, and duration. Individual tests are performed in order to prevent unauthorized access to program code.

The stress test is a method in which the software is exposed to conditions that exceed the normal operating conditions of the software. After reaching a critical point, the results are recorded. This test determines the stability of the entire system. The software is checked for compatibility with external interfaces such as operating systems, hardware platforms, web browsers, etc. The compatibility test verifies that the product is compatible with any software platform.

As the name suggests, this testing technique checks the amount of code or resources that the program uses to complete an operation.

This test checks the usability of the software for users. The ease with which the user can access the device is the main point of testing. Usability testing covers five aspects of testing - trainability, efficiency, satisfaction, memorization, and error.

8.3. Testing in white box

Testing in a white box, unlike black, takes into account the internal functioning and logic of the code. To perform this test, the tester must have knowledge of the code to determine the exact part of the code that has errors. This test is also known as White-box, Open-Box or Glass box testing.

Testing in a white box is a detailed study of the logic and structure of the code. Knowledge of the operation of the code is required for its implementation. The program code is inspected and checked which unit is not working properly. The method is suitable for testing applications providing web services and is not practical for debugging in large systems and networks. It is considered a security test, i.e. the process of determining whether the information system protects the data and maintains the functionality. The method can be used

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

to validate code implementation, i.e. whether it conforms to the design, to validate implemented security functionality, and to detect exploitable vulnerabilities.

9. Risks and testing;

Risk is the possibility of a negative or undesirable result or event. Each problem that arises reduces the perception of product quality or project success. "Risks are uncertain events that are likely to occur in the future and are likely to result in losses." Risk identification and management are fundamental concerns in any project. Effective software risk analysis helps to effectively plan and assign work tasks.

The risk is mainly expressed in two types:

- product (quality) risk - the primary effect of a potential problem on product quality;

- Project risk (planning) - the primary effect is on the success of the project.

The risks are not equal in importance. Determining the level of risk depends on several factors (Table 1.1 - Determining the risk according to the probability of occurrence and impact on the system):

- likelihood of occurrence - arises from technical considerations, such as programming languages used and Internet connection;

- impact of the problem in case of occurrence - arises from business considerations, such as financial losses and number of affected users. The effort is distributed in proportion to the level of risk, respectively the more important tests are conducted first.

For product risks we must take into account:

- which functions and attributes are critical for the success of the product;

- how visible to customers or consumers is a problem in one;

- how often a function is used; - Is it possible to skip this functionality?

Risk categories:

Schedule risk: The project schedule may be delayed when the project tasks and the deadline for delivery of the product are addressed correctly. Schedule risks can affect the project, and ultimately the savings the company makes can lead to project failure. Schedules often fail for the following reasons:

• Poor judgment of time.

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

• Inability to properly assess the functionalities and the time required to develop these functionalities.

• Resource use is not monitored properly. All resources such as staff, systems, individual skills.

• Unexpected extensions of the project scope.

Operational risks: Risks caused by lack of proper implementation, systemic failures or some external risk events.

Causes of operational risks:

• Inability to properly address conflicts in prioritization

• Failure to settle responsibilities

• Insufficient resources

• Insufficient team training

• Non-planning of resources

• Lack of communication in the team

Technical risks: Technical risks usually lead to failures in functionality and performance.

The causes of technical risks are:

• Constant change of requirements.

• The use of outdated or early-stage technologies.

• Complexity of the project implementation

• Complex modular project integration.

Pragmatic risks: These are external risks beyond operational borders. These are all unclear risks beyond the scope of the program.

These external events can be:

• Exhaustion of funds

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

• Market development

• Changing the product strategy and priority

• Change in state regulations. Budgetary risks:

• Poor budget assessment

• Overspending

• Extending the scope of the project

Thematic cycle 2: “Software development”

I. Basics of programming:

What does "programming" mean?

To program means giving commands to the computer what to do, such as "play a sound," "print something on the screen", or "multiply two numbers". When there are several commands in a row, they are called a **computer program**. The text of computer programs is called **program code** (or **source code** or for short **code**).

Computer programs

Computer programs are a **sequence of commands** that are written in a pre-selected **programming language**, such as C++, Java, JavaScript, Python, Ruby, PHP, C, C#, Swift, Go, or another. To write commands, we need to know the **syntax and semantics of the language** we will be working with, in our case C++. Therefore, we are going to get familiar with the syntax and the semantics of the language C++, and with programming generally, in the current book, by learning step by step code writing from the simpler to the more complex programming structures.

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

Algorithms

Computer programs usually execute some algorithm. **Algorithms** are a sequence of steps, necessary for the completion of a certain task and for gaining some expected result, something like a "recipe". For example, if we fry eggs, we follow some recipe (an algorithm): we warm up the oil in a pan, break the eggs inside it, wait for them to fry and move them away from the stove. we warm up the oil in a pan, break the eggs inside it, wait for them to fry and move them away from the stove. Similarly, in programming the **computer programs execute algorithms**: a sequence of commands, necessary for the completion of a certain task. For example, in order to arrange a series of numbers in ascending order, an algorithm is needed, for example to find the smallest number and print it, to find the smallest number again from the other numbers and print it, and this is repeated, until the numbers run out.

For convenience when creating programs, for writing programming code (commands), for execution of programs and other operations related to programming, we need a **development environment**, for example Visual Studio.

Programming languages, compilers, interpreters and development environments

A **programming language** is an artificial language (syntax for expression), meant for **giving commands** that we want the computer to read, process and execute. Using programming languages, we write sequences of commands (**programs**), which **define what the computer should do**. The execution of computer programs can be done with a **compiler** or with an **interpreter**.

The **compiler** translates the code from programming language to **machine code**, as for each of the structures (commands) in the code it chooses a proper, previously prepared fragment of machine code and in the meantime it **checks the text of the program for errors**. Together, the compiled fragments comprise the program into a machine code, as the microprocessor of the computer expects it. Once the program has been compiled, it can be executed directly from the microprocessor in cooperation with the operating system. With compiler-based programming languages the **compilation of the program** is done obligatory before its execution, and syntax errors (wrong commands) are found during compile time. Languages like C++, C#, Java, Swift and Go work with a compiler.

Some programming languages do not use a compiler and are being **interpreted directly** by a specialized software called an "interpreter". The **interpreter** is "**a program for executing programs**", written in some programming language. It executes the commands in the program one after another, as it understands not only a single command and sequences of commands, but also other language constructions (evaluations, iterations, functions, etc.). Languages like Python, PHP and JavaScript work with an interpreter and are being executed without being compiled. Due to the absence of previous compilation, in interpreted languages the **errors are being found during the execution time**, after the program starts running, not previously.

The **programming environment** (Integrated Development Environment – **IDE**) is a combination of traditional tools for development of software applications. In the development environment we write code, compile and execute the programs. Development environments

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

integrate in them a **text editor** for writing code, a **programming language, a compiler or an interpreter and a runtime environment** for executing programs, a **debugger** for tracking the program and seeking out errors, **tools for user interface design** and other tools and add-ons.

Programming environments are convenient, because they integrate everything necessary for the development of the program, without the need to exit the environment. If we don't use a development environment, we will have to write the code in a text editor, to compile it with a command on the console, to run it with another command on the console and to write more additional commands when needed, which is very time consuming. That is why most of the programmers use an IDE in their everyday work.

For programming with the **C++ language** the most commonly used development environment is **Visual Studio**, which is developed and distributed freely by Microsoft and can be downloaded from: https://www.visualstudio.com/downloads/. Alternatives of Visual Studio are **Rider** (https://www.jetbrains.com/rider/), **MonoDevelop** / **Xamarin Studio** (http://www.monodevelop.com), **SharpDevelop** (http://www.icsharpcode.net/OpenSource/SD/) and **Eclipse** (https://www.eclipse.org/downloads/packages). Another common development environment is **Code::Blocks** (http://www.codeblocks.org/downloads).

Variables and expressions

In programming **variable** is a place to store some value in computer RAM. It is indicated by the programmer with a name (identifier), usually in Latin. Since a value with information in a computer can be of a given type (integer, fraction, letter, structure, object, etc.), the variable that contains it is said to be from the corresponding type: integer variable, symbolic, object, etc.

The name of the variable indicates a specific area in memory where its value is contained. This separation of name and content allows the name to be used regardless of the information it represents. Variables can be stored directly in the program's working memory (stack) or in dynamic memory, in which larger objects (character strings and arrays) are stored. Primitive data types (numbers, char, bool) are called value types because they store their value directly in the program stack. Reference data types (strings, objects, and arrays) store an address in dynamic memory where their actual value is stored. They can be set aside and released dynamically, i.e. their size is not fixed in advance, as with value types. The identifier in the code can be set while the program is running. However, during its execution, this value may change. Variables can be used in different processes: to set a value in one place, then in another, then they can accept a new value and use it in the same places where it was used before. The value of the variable can be changed during program execution, as long as its name, type, and memory location remain unchanged.

Variables in programming most often have descriptive names, depending on what they contain and how they can be used, as opposed to variables in mathematics, which usually consist of 1-2 characters. When we want the compiler to allocate an area in memory for some information used in our program, we have to name it. The symbols that are allowed are the letters a-z, A-Z, the numbers 0 - 9, as well as the symbol '_'. We must also take into account that programming languages have official words and we cannot use them. The name serves as

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

an identifier and allows us to refer to the required area of memory. When we declare a variable, we perform the following actions:

- set its type (for example int);

- set its name (identifier, for example age);

- we can set an initial value (for example 25), but this is not mandatory.

The syntax for declaring variables in C# and Java is as follows:

```
<data type> <identifier> [= <initialization>]
```

Here is an example of declaring variables:

```
string name;
```

```
int age;
```

Assigning a value to a variable is setting a value to be written to it. This operation is performed via the assignment operator '='. The name of a variable is displayed on the left side of the operator, and its new value is displayed on the right side. Variable initialization means the first setting of a value.

Much of the work of a program is the calculation of **expressions**. Expressions are sequences of operators, literals and variables that are calculated to a certain value of some type (number, string, object, or other type).

Calculating an expression can also have side effects, because the expression can contain built-in assignment operators, increasing or decreasing value (increment, decrement) and calling methods.

Conditional code

In computer science, conditional structures are functions of programming language, through which we can perform various actions depending on some condition.

In imperative programming languages, the term "conditional structures" is usually used, while in functional programming, the terms "conditional expression" or "conditional structure" are preferred because these terms have different meanings.

Boolean expressions

The set of Bool type values (bool in C/C++/C#, boolean in Java) consists of two elements - the values true and false. These values are also called Boolean constants.

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

Boolean expression in computer science we call any expression which, after its transformation, can be reduced to a Boolean constant.

x	y	! x	x && y	x || y	x ^ y
true	true	false	true	true	false
true	false	false	false	true	true
false	true	true	false	true	true
false	false	true	false	false	false

Comparison operators

In high-level languages, we distinguish several types of comparison operators that are used to compare pairs of integers, floating-point numbers, strings, and other data types. All expressions that contain comparison operators are Boolean expressions because their result is reduced to a Boolean constant.

Operator	Action
==	equal
!=	different
>	greater
<	less
> =	greater or equal
<=	less or equal

Logical operators

Logical operators accept Boolean values and return a Boolean result.

Logical operator	Stylized record	Name
&&	AND	AND
||	OR	OR
!	NOT	Logical negation
^	XOR	Exclusive OR

Types of conditional structures

In the languages of the C family we distinguish the following types of conditional structures:

if / else / else if / embedded if structures

A program can behave differently, depending on a condition, using the conditional if and if-else structures. They are a type of conditional control that is checked during the execution of the structure.

if

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

The format of the conditional if structure is as follows: if-clause, Boolean expression and body of the conditional structure;

```
if (Boolean expression)
{
 body of the conditional structure;
}
```

Boolean expressions can be Boolean type variables or Boolean Boolean logical expressions. In C# they cannot be an integer unlike other programming languages such as C and C++.

The body of the structure is locked between the large curly brackets "{}" and can consist of one or more operations (statements).

If the expression in brackets after the if keyword is calculated to true, the body of the conditional structure is executed. If the result of the Boolean expression calculation is false, then the operators in the body will not be executed.

else

There is also a conditional structure with an else clause (if-else). Its format includes: reserved word if, Boolean expression, body of conditional structure, reserved word else and body of else-structure, which may consist of one or several operators:

```
if (Boolean expression)
{
 body of the conditional structure;
}
else
{
 body of else-structure;
}
```

Here, this structure works as follows: depending on the result of the expression in brackets (Boolean expression), two paths are possible to continue the flow of calculations. If the Boolean expression is true, the body of the conditional structure is executed, and else is omitted as the operators in it are not executed. Otherwise, the else structure is executed, but the main body is omitted and the operators in it are not executed.

else if

In addition to if and else, which can describe 2 cases in total, we can have else if structures that check several conditions. It is mandatory that else if structure be used after an if structure, and after else if, there can be a single else structure, but this is optional.

```
if (x == 0)
```

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

```
{
 Console.WriteLine("The number is 0");
}
else if (x == 1)
{
 Console.WriteLine("The number is 1");
}
else if (x == 2)
{
 Console.WriteLine("The number is 2");
}
else
{
 Console.WriteLine("Another number");
}
```

As we already know, if checks some condition, if this condition is met, we go to the execution of the code in this if structure and ignore the code of all other else if and the optional else structure. If the condition in if is not met, we check the conditions of each else if in the order in which they are written. When a match is found, the corresponding code is executed and the others are ignored. If there is an else structure and we did not have a match in if or else if, then the code in the else structure is executed, which covers all cases not covered by the previous structures.

Embedded if structures

Embedded if or if-else structures often find application in program logic in a program or application. This is done by placing an if or if-else structure in the body of another if or else structure. Here, each else clause refers to the closest previous if clause. This way we understand which else clause to which if clause refers.

No more than three conditional structures must be embedded, otherwise part of the code must be exported in a separate method.

switch-case

The conditional structure switch (multiple branching in some programming languages) is used to select from a list of options. The structure compares a given value with certain constants and based on the comparison with any of them, takes an action.

```
switch (selector)
{
 case value-1: execution code; break;
 case value-2: execution code; break;
 case value-3: execution code; break;
 case value-4: execution code; break;
```

```
// ...
default: execution code; break;
}
```

The switch-case structure chooses from parts of program code based on the calculated value of a certain expression. This expression is usually integer, but can also be of string or char type. The value of the selector must be calculated before comparing with the values inside the switch structure. Tags (case) do not have to have the same value. When the selector matches any of the case values, the switch-case structure executes the code after the corresponding case. In the absence of a match, the default structure is executed, when such exists. Each case tag, as well as the default tag, must end with the break keyword, which terminates the switch-case structure once a match is found and the corresponding code is executed.

In C Sharp we have the ability to use multiple tags when they need to execute the same code. In this way of writing, when we find a match, because after the corresponding case tag there is no execution code and break operator, the next encountered code will be executed. If such is missing, the default structure will be executed.

```
int number = 6;
switch (number)
{
 case 1:
 case 4:
 case 6:
 case 8:
 case 10:
 Console.WriteLine("This is not a prime number!"); break;
 case 2:
 case 3:
 case 5:
 case 7:
 Console.WriteLine("This is a prime number!"); break;
 default:
 Console.WriteLine("I don't know what that number is!"); break;
}
```

Good practices in the use of switch-case

- It is good practice to always use a default structure to be placed at the end, after the other case tags, to handle atypical values that the selector may accept or even situations that may be considered incorrect.
- It is good to put in the first place those case cases that deal with the most common situations, and to leave the case structures that deal with less frequent situations at the end.

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

- If the values in the case tags are integers, it is recommended that they be arranged in ascending order.
- If the values in the case tags are of the character type, it is recommended that the case tags be arranged alphabetically.

A ternary operator?

The conditional operator ? : is an operator in the C language and C-like languages. It is also known as a ternary operator, as it is the only operator that accepts 3 operands.

```
operand1 ? operand2 : operand3
```

The first operand or the condition of the conditional structure can be Boolean variable or Boolean expression and can accept both Boolean values true and false. If after performing the necessary transformations operand1 is reduced to true statement, then after its execution the ternary operator will return the value of operand2, otherwise (false), the returned value will be the value of operand3.

```
int a = 5;
int b = 3;
int larger = (a > b) ? a : b;
```

In the example above we initialize 2 integer variables a and b and give them values of 5 and 3. For the variable larger we assign the result by the ternary operator. In this case it is 5, since the condition (a > b) is fulfilled (true), the returned value will be that of the operand before the two points, i.e. the variable a, which is 5. This example shows how we can easily determine which of the two numbers is greater without using the if-else structure. The same example implemented with if-else would look like this:

```
int a = 5;
int b = 3;
int larger;
if (a > b)
 larger = a;
else
 larger = b;
```

Loops and iteration

The **cyclic calculation process** is process, for short called a loop, which is the repeated execution of a sequence of operations with different data. Most often, only one value is changed, which is called a loop parameter. For different types of loops, the program code is

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

repeated until a predetermined condition or a fixed number of times is in force, which are mentioned at the beginning.

Each cyclic process is characterized by the following elements:

- Initialization - sets the initial value of the loop parameter.
- Loop body - initializes the code that must be executed a certain number of times.
- Update - the value of the loop parameter is updated.
- Interrupt condition - the expression, depending on which value the loop stops or continues its action.

Omission or incorrect setting of any of the elements of the loop can lead to an error in its execution or inability to execute.

For loop

The name for comes from the English word for - translated into Bulgarian - "for", because during its execution the body of the loop is executed for a variable in the loop itself. The word for is used as a keyword in most programming languages.

In computer science, for loops are blocks of program code that can be executed by repeating the operations entered in them.

Unlike other types of loops, a counter is introduced in the structure of for loops, through which the number of iterations can be controlled. For loops are applicable when it is predetermined how many iterations the program should perform.

Structure

The structure of for loops includes:

a) initialization block

b) loop condition

c) commands for updating the leading variables

d) Body of the loop

Example of presenting the structure of for loop in the language C#:

```
for (initialization-„a”; condition-„b”; update on leading variable-”c”)
{
 Body of the loop - “d”
}
```

Body of the loop

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

The body of the loop contains the real part of the program - a block with source code (also called source code or program code). The variables declared in the initialization block of the loop are available in it. This code is executed at each subsequent iteration (repetition) until the condition to terminate the loop is met and exits the loop.

Examples

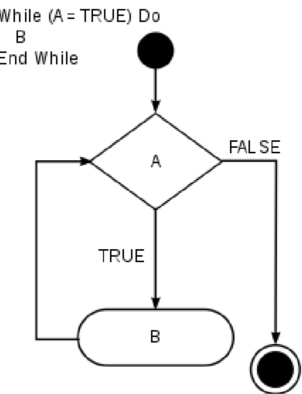
Examples of the implementation of for loop in C# programming language:

```
for (int i = 2; i <= 20; i += 2)
{
 Console.Write(i + " "); //For each iteration, the next even number + one space is printed.
}
// Result: 2 4 6 8 10 12 14 16 18 20
```

In this example, 10 iterations are performed in a for loop. Before each of them, it is checked whether the condition for termination of the loop - the value of the variable i, is greater than 20. Until this condition is met - the variable i has a value less than or equal to 20, all even numbers that are greater than 1 and less than or equal to 20 are printed.

The while loop

The while loop is a loop that is executed as long as a predefined Boolean condition at the beginning of the loop is true.



Structure of while loop

The loop works as follows:

1. The condition is checked and if it is true it continues to the code in the body of the loop. Otherwise, the loop ends.
2. The code in the body of the loop is executed.
3. The loop condition is checked again, etc.

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

The body of the loop must be enclosed in curly brackets ({body of the loop}) and the condition in square brackets (condition).

Example of while loop in C#

```
int i = 0;
 while (i<10) //condition
 {
 Console.Write(i); //body of the loop
 i++;
 }
// result 0123456789
```

A variable i is set that is equal to 0. The condition of the loop is to work as long as i is less than 10. With each rotation of the loop i increases by 1. Otherwise the loop would be infinite and will print 0 on the screen every time. At the last rotation of the loop, it prints 9 on the console and increases the value of i by one. Now i is equal to 10. When checking the condition at the beginning of the loop, it is already false, because i is not less than 10, the loop interrupts its work.

Do-while loop

The do-while structure is similar to the while loop, but with the difference that the condition is set at the end, i.e. the check is made after the body of the loop is completed.

Example of a do-while loop in C#

```
int i = 0;
 do
            {
             Console.Write(i); //body of the loop
 i++;
            }
 while (i < 10) //condition
```

Embedded loops

If a new loop instruction is included in the body of a loop, a more complex structure called "loop in loop" or embedded loops is obtained. In this case, the loop that is in the first is called internal, and the first loop is called external. Each of the two loops is characterized by the four main elements - initialization, body, interrupt condition and update. It is mandatory that the parameters of the two loops are different variables, i.e. to be named differently. In each execution of the actions in the external loop, a full number of executions of the actions in the internal loop are performed.

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

Example of embedded loops in C#

```
for (int i = 0; i < 3; i++)
{
 for (int j = 0; j < 5; j++)
 {
 Console.Write(j);
 }
 Console.WriteLine();
}
// result
// 01234
// 01234
// 01234
```

Indefinite loop

A loop that never ends is called an infinite loop . An infinite loop is a series of commands in computer program, which performs an infinite number of iterations. Infinite loops can occur in for, while, do-while loops. The reasons for their occurrence may be:

- Due to the lack of a condition for termination of the program.
- A condition has been introduced to terminate the loop, which can never be fulfilled.
- A condition has been introduced for termination of the loop, after the execution of which, the loop starts working from its starting point.

2. Algorithms, elements of C/C ++ programming languages, basic data types;

C++ is a general-purpose programming language. It was created by Bjarne Stroustrup, and work on the project began in 1979, and the first public version of the language was published in 1985. The name C++ is related to the principles around which Stroustrup built his language. He took the C language as a basis and tried to add the concept of classes to it. This is how the "C with classes" project was born. A long evolution followed, eventually giving rise to the C++ language.

The main feature that distinguishes it from its predecessor is the possibility of object-oriented programming (OOP). OOP is a paradigm in which program code is divided into classes, objects and methods that store the necessary information and communicate with each other, if necessary. This programming concept is the opposite of the concept of procedural programming, in which the program is sequentially executed instructions (computational steps). OOP introduces a more abstract approach to program code, which allows programmers to focus more on program logic.

However, at a more abstract level, C++ retains enough properties that are familiar to lower-level languages. One of them is memory management. This is a key element in C++ programming because, unlike higher-level languages (where this activity is done

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

automatically). This feature (for manual memory management) makes C++ a language that is suitable for programs that will run in a hardware environment with a more limited resource.

Types of data

A variable in C ++ must specify a data type:

Example

```
int myNum = 5; // Integer (whole number)
float myFloatNum = 5.99; // Floating point number
double myDoubleNum = 9.98; // Floating point number
char myLetter = 'D'; // Character
bool myBoolean = true; // Boolean
string myText = "Hello"; // String
```

Basic data types

The data type determines the size and type of information that the variable will store:

Data Type	Size	Description
int	4 bytes	Stores whole numbers, without decimals
float	4 bytes	Stores fractional numbers, containing one or more decimals. Sufficient for storing 7 decimal digits
double	8 bytes	Stores fractional numbers, containing one or more decimals. Sufficient for storing 15 decimal digits
boolean	1 byte	Stores true or false values
char	1 byte	Stores a single character / letter / number, or ASCII values

Numeric types

Use int when you need to store an integer without decimal places, such as 35 or 1000, float or double when you need a floating point number (with decimal places), such as 9.99 or 3.14515.

int

```
int myNum = 1000;
cout << myNum;
```

float

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

```
float myNum = 5.75;
cout << myNum;
```

double

```
double myNum = 19.99;
cout << myNum;
```

float vs. double

The floating point **accuracy** value indicates how many digits the value can have after the decimal point. The accuracy of float is only six or seven decimal places while double variables have an accuracy of about 15 digits. Therefore, it is safer to use double for most calculations.

Scientific numbers

The floating point number can also be a scientific number with "e" to indicate the strength of 10:

Example

```
float f1 = 35e3;
double d1 = 12E4;
cout << f1;
cout << d1;
```

Boolean types

Boolean data type is declared with bool keyword and can only accept values true or false. When the value is returned, true = 1 and false = 0.

Example

```
bool isCodingFun = true;
bool isFishTasty = false;
cout << isCodingFun; // Outputs 1 (true)
cout << isFishTasty; // Outputs 0 (false)
```

Boolean values are mostly used for conditional testing.

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

Types of symbols

The type char data is used to store a single character. The character must be surrounded by single quotes, such as "A" or "c":

Example

```
char myGrade = 'B';
cout << myGrade;
```

Alternatively, you can use ASCII values to display certain symbols:

Example

```
char a = 65, b = 66, c = 67;
cout << a;
cout << b;
cout << c;
```

String types

The string type is used to store a sequence of characters (text). This is not an embedded type, but it behaves as such in its most basic use. Standing values must be enclosed in double quotes:

Example

```
string greeting = "Hello";
cout << greeting;
```

To use strings, you must include an additional header file in the source code, <string>library:

Example

```
// Include the string library
#include <string>

// Create a string variable
string greeting = "Hello";

// Output string value
cout << greeting;
```

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

3. Consistent and conditional execution, iterative solutions, functions;

Consistent and conditional execution

In programming, we have a special group of instructions that violate the conceptual order of sequential execution in Neumann architecture. These instructions include the instruction for conditional transition (**branch**), unconditional transition (**jump**),

Aspects of control instructions

- There will be / There will be no transition.
- How to embed (set) the condition in branch (often the condition checks the flag register). The flag register is determined by the operations performed by the ALU. The flag changes with each operation.
- How to calculate the transition address.
- Link Return Address - binds return addresses (done in transitions **calls** and **returns** - i.e. in subprograms)

Control instructions

- **jump** for unconditional transition;
- **branch** for conditional transition

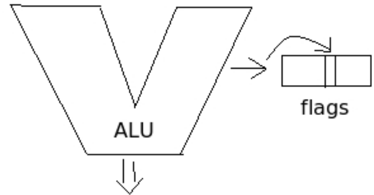
Details

jump / branch

jump always changes the contents of the program counter, while **branch** has a condition and PC changes only if this condition is met. How the branch condition will be embedded is a matter of architecture implementation. Conditional transitions can be made in such a way that the value of the condition is calculated in them. Most often the condition is a comparison - a transition is performed if the result of the comparison is positive, so most often the branch instructions describe what is being compared and the address of the transition. The comparison is performed in the ALU, this is one type of architecture with conditional transitions with comparison instructions. In the other type of architectures, the comparison instructions are separated from the conditional transition instructions and they are called immediately before.

flags

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.



The condition often tests a flag register. Signs of the result are recorded in the flag register. Each arithmetic operation puts some flags. These are special rules by which the transition condition is recognized, depending on the state of the flags.

A disadvantage of the conditional transition codes is that they provide an implicit connection between the instructions - for example, a conditional transition must be performed after performing an arithmetic operation with ALU for numbers with a fixed point. In pipelining (conveyor) this can disrupt the speed - in the transition instructions the whole conveyor is cleared.

Formation of the transition address

The transition address is formed in 4 possible ways:

1. as a relative address to the program counter (PC + offset);
2. By basic register and displacement (Rbase + displacement);
3. By setting an absolute address (special case of the above at Rbase == 0);
4. Through vectors.

Saving registers

The last aspect of the control instructions is the saving of the registers. To run faster, a program works with operands written to registers. With each transition to a subprogram, this context is lost and must be restored upon return. This program must have a **save area** in which this context is temporarily saved. The software convention is the calling function to take care to keep its context, and to restore it when returning from the subprogram. The return location must have instructions for returning the context, i.e. **multiple load** and **multiple store** are made each time, to save and load all registers. There is a Rsa register in which the address of this **save area** is kept. There is another convention (**callee**) in which the subprogram takes care before return to fill the registers with the information, as it was when it was called.

Groups of registers

- in - the arguments accepted by a subprogram
- out - the result of the execution (it can be returned to the calling program)

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

- local - local variables
- global - global variables

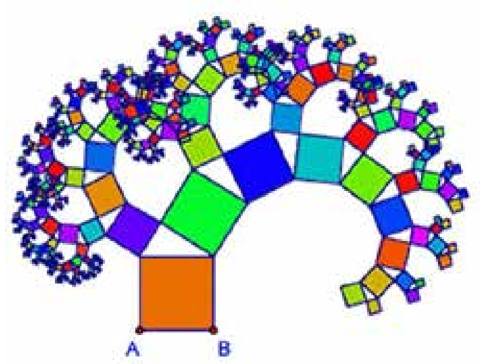
When entering, save in and local, and when leaving out and local. For system calls, all registers are kept, and for regular calls only the basic ones.

Iterative solutions

Introduction to the term "iteration"

Iteration (in Latin iterare "repeat") is a term generally used to indicate "the repetition of a process". Iteration is a step in a process that is most commonly used to solve quantitative problems and that is repeated several times. Each repetition is called an iteration.

In programming, most often iteration means "loop" (return to a condition until it becomes true and the program continues).



The Fractal Pythagorean Tree

Fractal imaging is closely related to programming. For people who have even a vague idea of programming, these concepts are clear. For the others:

An iteration in programming is an organization of data processing in which the action is repeated several times without leading to a situation in which the function calls itself (this is already recursion).

In general, an iteration can be translated as "repetition".

Each fractal has an infinitely repeating form. When creating such a fractal, the simplest way is to repeat several actions that create this form. Instead of the word "repetition" we use "iteration". Virtually any fractal can be created with iterations of some rule. To create a real fractal, you have to iterate an infinite number of times. But we could do it a finite number of times, but long enough to get an idea of the "real" fractal. Naturally with the help of a computer. Increasing the number of iterations makes fractals more accurate.

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

There are three main types of iterations:

1. **Substitute iteration** - creates fractals, replacing some geometric shapes with others.

We start with a figure called a **base**. Then we replace each part of the base with another figure called a **motif**. We perform this substitution an infinite number of times until we complete the fractal.

L-systems

The replacement iteration is very simple. But for a computer, it is not enough to have the image of the base and the motif. A clear and accurate way to store fractal data is needed to allow simple algorithms to be drawn to draw fractals.

L-systems are a good and simple way. They were developed by A. Lindenmeier ("L" in the word "L-system"). They consist of an angle, an axiom and at least one rule. We call an **axiom** the initial form (base) that will be used in the process of creating a fractal. The rules specify which symbols in the axiom should be replaced by other symbols.

<table>
<tr><td colspan="6">SYMBOLS</td></tr>
<tr><td colspan="2">Common symbols</td><td colspan="2">Complex symbols</td><td colspan="2">Coloring symbols</td></tr>
<tr><td>F</td><td>Moving forward, with a trail</td><td>@n</td><td>Multiply the segment length by n</td><td>Cn</td><td>Determine color number n</td></tr>
<tr><td>G</td><td>Moving forward without a trail</td><td>I</td><td>In front of a number means division instead of multiplication</td><td><n</td><td>Decrease the color number by n</td></tr>
<tr><td>+</td><td>Rotate counterclockwise</td><td>Q</td><td>In front of a number it gives a square root</td><td>> n</td><td>Increase the color number by n</td></tr>
<tr><td>-</td><td>Rotate clockwise</td><td>!</td><td>Changes the sign from + to -</td><td colspan="2" rowspan="3"></td></tr>
<tr><td>|</td><td>Rotate 180°</td><td>[</td><td>Insert the current cursor position into memory</td></tr>
<tr><td colspan="2"></td><td>]</td><td>Remove the last saved cursor position from memory</td></tr>
</table>

Most fractals with a fractal dimension of 0 to 2 can be expressed using L-systems. With a combination of several symbols and rules, very complex fractals can be created. Such L-systems are used to create **realistic plant models**.

It is also possible to achieve greater realism by introducing a parameter that adds random numbers. The larger their range, the more natural the forms will look and away from symmetry.

2. **Iteration IFS** - creates fractals, applying geometric transformations (type of rotation and reflection).

IFS (iterative functional systems) represent another way to create fractals. This method is based on a point or figure, which is replaced by several smaller figures.

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

3. **Iteration with formulas** - includes several ways to create fractals, repeating some mathematical formula or several formulas.

Formula iteration is the simplest type of iteration, but it is the most important and gives the most complex results. Algebraic fractals are constructed by formula iteration, i.e. by mathematical formulas. We will discuss them in the topic of Algebraic Fractals.

Functions

Functions are the basic structural units from which programs are developed. Each function consists of multiple operators (maybe 0), which are executed as one generalized operation or action. Once a function is created, which is accomplished by defining it, it can be executed repeatedly. Each execution of a function can have different data, as the specific data is set when calling /activating/ the function. As a result of their execution, functions can return a value called the return value.

Each program consists of 1 or more functions. Only one of the functions in the program is named main. The main function is the first function that is executed when the program is started. Good programming style requires developing programs from many small functions, which:

- makes programs clearer, easier to test, set up and modify;
- the repetition of the same fragments of the programs is avoided. These fragments can be defined once as functions and then executed repeatedly;
- memory savings are achieved because the code of a function is stored only in one place in the memory, regardless of the number of its executions.

The functions are of two types:

- a function that returns a value of some type /true, real/ function /it is called as an operand in an expression and is equivalent to one multiplier/;
- function that does not return a value /procedure in Pascal/ - it is called as an independent operator in another function;

4 Arrays, matrices and their applications;

An array is an ordered sequence of elements of the same base type. An individual element of the array is indicated by the name of the whole array, followed by the sequence number (index) of the element. In different programming languages, arrays are built in different ways.

Characteristics of the arrays

The array can be one-dimensional, multidimensional or an array of arrays.

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

They are based on zero indexing - this means that in an array with N elements, the first element will have an index of zero, and the last with an index of N-1.

Array elements can be of any type, including array type.

The default value of numeric type elements is zero, for reference types it is null, and for Boolean types it is false. In an array of arrays, the elements are of the reference type and are null by default.

The order of the elements and the length of the array are fixed.

Multidimensional arrays

For a two-dimensional array, the elements with indices i, j would have the address B + ci + dj, where the coefficients c and d are the incremental steps of the row and column, respectively.

More generally, in a k-dimensional array, the address of an element with indices i1, i2,…, ik is:

$B + c1 \cdot i1 + c2 \cdot i2 + \ldots + ck \cdot ik$.

For example: int a[3][2];

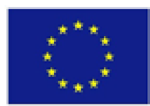
This means that the array is of type int and has 3 rows and 2 columns. In it we can store 6 elements, arranged linearly, starting from the first row. The above array was stored in the following order: a11, a12, a13, a21, a22, a23.

This formula requires only k multiplications and k additions for any array that can be stored in memory. In addition, if the coefficient is an exact exponent of 2, the multiplications can be replaced by Bit Shifts.

The coefficients ck must be such that each valid index leads to the address of a certain element.

If the minimum allowed value for each index is 0, then B is the address of the element on which all indices are 0. As with one-dimensional arrays, the indices of the elements can be

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.



changed by changing the base address B. Therefore, if a two-dimensional array has rows and columns with indices from 1 to 10 and from 1 to 20, respectively, then replacing B with B + c1 - - 3 c1 we will renumber the indices of the elements to 0 to 9 and 4 to 23, respectively. Taking advantage of this property , some programming languages (such as FORTRAN 77) set array indices to start at 1, as is traditionally mathematical, while other languages (such as Fortran 90, Pascal, and Algol) allow the user to select the minimum value of each index.

Processing of arrays

with for loop - used when working with the index of the elements and it is not necessary to crawl each of the first to the last element.

with foreach loop - used when the use of an index is not necessary and the elements are crawled one by one. With this operation, the elements cannot be changed, only read.

Arrays as objects

In C# language, arrays are actually objects. System.Array is the basic type of all types of arrays and its characteristics can be applied. An example of this is finding the length of an array using the System.Array.Length method.

Scalable arrays

These are arrays that can be changed dynamically by adding or removing elements from them. The syntax is List <T> - where T is the type of data that will be contained. Their main advantage is that we do not need to know in advance the length of the array. Initially, the newly created list has 0 elements. Basic methods and properties:

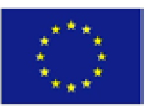
: Add (T element) - adds an element at the end

: Remove (element) - removes the element

: Count - returns the current length of the list

Arrays in C/C++

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.



In the C and C++ programming languages, an array is a group of variables that are of the same data type, have the same size, are sequentially arranged in memory, and there is a pointer that points to the first element of the array.

Arrays in PHP

In the PHP programming language, an array is a variable that contains many elements. Arrays can be of 2 types - ordinary and associative. For ordinary arrays, an index (sequence number) of the element is used to refer to an element. In associative arrays, a text string is used to reference an element.

Arrays in JavaScript

In the JavaScript programming language, an array is an object. It has a constructor and methods. Indexes or text strings can be used to refer to the elements of the object.

Arrays in Python

In the Python programming language, arrays are of 2 types - tuple and list. The difference between them is that a tuple cannot be changed - as long as it exists, its elements remain as they were when it was created. In list, the elements can be changed.

Arrays in C#

Arrays in C# language are a collection of several variables of the same type. These are called array elements.

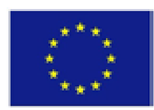
Matrices

A two-dimensional array is also called a matrix that has set columns

There are two types of matrix:

1.Square matrix where the columns are equal to the rows
2.Rectangular matrix where the columns are NOT equal to the rows

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.



When working with a matrix in programming, a two-dimensional array (rows and columns) is used, which is most often crawled with two embedded for loops.

Types of matrices:

- square matrix - the number of rows is equal to the number of columns
- symmetrical matrix - a square matrix which elements are symmetrically located about the main diagonal are equal
- triangular matrix - a square matrix in which all elements below or above the main diagonal are zeros, respectively, upper or lower triangular matrix;
- diagonal matrix - a square matrix which non-zero elements are only in the main diagonal;
- scalar matrix - diagonal matrix, all elements of the main diagonal are equal
- single matrix - a scalar matrix with elements of the main diagonal equal to one

5. Strings processing elements;

The string in computer science is a finite sequence of symbols (representing a finite number of characters). Its elements can be changed, as well as its length. Strings are usually treated as a data type and are often converted into arrays of bytes (or words) that store a sequence of elements using symbol encoding.

Depending on the programming language and the data type used, a variable declared as a string can be stored in memory statically (predefined maximum length) or dynamically (it can contain a different number of elements).

In the formal languages used in mathematical logic and theoretical computer science, a string is a finite sequence of symbols from a set called an alphabet.

String length

Although strings can be arbitrary (but limited) in length, in most programming languages the length is limited to an artificial maximum. There are generally two types of strings:

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

- with a fixed length - they have a constant maximum length and use the same amount of memory, regardless of whether this maximum is reached;
- with variable length.

Formal theory

Let us indicate by Σ the alphabet, which is an empty finite set. The elements of Σ are called letters. A string (or word) of Σ (Σ*) is any finite sequence of letters of Σ. For example, if Σ = {0, 1}, then 0101 is a string of the alphabet Σ.

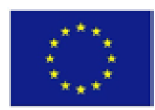
The set of all strings over Σ of length n is written Σn. Example: if Σ = {0, 1}, then Σ2 = {00, 01, 10, 11}. Note that Σ0 = {ε} for each alphabet Σ.

A set of strings over Σ (for example, any subset of Σ*) is called a formal language over Σ. Example: if Σ = {0, 1}, the set of strings with equal number of zeros and ones ({ε, 1, 00, 11, 001, 010, 100, 111, 0000, 0011, 0101, 0110, 1001, 1010, 1100, 1111, ...}) is a formal language over Σ.

Symbol encoding

In the .NET Framework, each symbol has a sequence number from the Unicode table. The Unicode standard was created in the late 80's and early 90's to store different types of text data. Its predecessor ASCII allows the recording of only 128 or 256 symbols (respectively ASCII standard with 7-bit or 8-bit table). Unfortunately, this often does not meet the needs of the user - as 128 symbols can only fit numbers, lowercase and uppercase Latin letters and some special symbols. When it is necessary to work with text in Cyrillic or another specific language (for example, Asian or African), 128 or 256 symbols are extremely insufficient. That's why .NET uses a 16-bit symbol code table. With the help of our knowledge of number systems and the presentation of information in computers, we can assume that the code table stores 2 ^ 16 = 65536 symbols. Some of the symbols are encoded in a specific way, so it is possible to use two symbols from the Unicode table to create a new symbol - the resulting characters exceed 100,000. String data types have used one byte per symbol in the past. Some languages such as Chinese, Japanese and others (also known as CJK) need much more than 256 symbols. One solution to this problem is to preserve single-byte ASCII representation and use double-byte for CJK languages. This

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.



leads to problems with string matching, length losses, and more. With Unicode, things are greatly simplified in this regard, and most programming languages support it.

The main elements of string processing are:

- Assignment operation
- Access to individual symbols
- Comparison of symbol strings
- Extract part of a string
- Function for merging symbol strings
- Function for determining the length (number of symbols) of a symbol string
- Procedure for converting a symbol string to a numeric value
- Procedure for converting a number to a symbol string

6. Structures

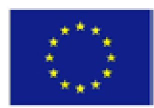
This is a special type that we create by specifying exactly what information (what data types and how many variables) it contains.

For example, often in geometric problems we have to keep points (say in three-dimensional space). To keep N such points, many competitors would use three arrays double x [N], y [N], z [N], and in each array they would keep one of the coordinates of the points. There are many situations in which this easy way would lead to more inconvenient code. For example, to pass a point to a function, we must pass each of its coordinates separately. Another option is to submit its index, and then the arrays must be global. An even bigger problem is if we want to rearrange the points.

We usually use a structure when we just want to encapsulate more complex information, and a class when we want to have additional methods that do some action on that data.

Instead, however, we can create a structure that represents a point for us. It would be struct Point {double x, y, z;} ;, and we would need only one array (instead of three): Point points [N] ;. Passing a point to a function is done by passing a single variable of type Point. Rearranging the points (for example when sorting) becomes much easier, but we'll talk about that later in this topic.

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.



The structures provide much more freedom. For example, there is no problem in storing different types of variables in it. If we want to have a guy who is a student, we might want to know his or her first name, last name, PIN, grade point average, and class number. Elements of a composite type (structure or class) are called "objects". Hence the name "Object-Oriented Programming".

The structure allows grouping of data of different types in one logically connected unit. In this the structure differs from an array, which is a set of elements of the same type.

Use of data structures

Most often, data structures are used as part of an algorithm - depending on the choice of data structure, it is differently fast. Some racing tasks are created specifically for an interesting data structure, requiring only its properties.

Basic types of structures

Linear

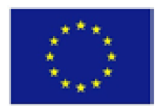
Linear data structures are lists, stacks, and queues.

A linear list is a series of elements of the same type. The main operations that can be performed with the elements are addition and removal.

Types of linear structures

- linear single-linked list - each item, except the last, is linked to the next with a single link. The list is crawled from beginning to end.
- linear double-linked list - each element, except the last, is connected to the next by two links. This simplifies operations. For example, a list item is easily accessible depending on whether it is closer to the beginning or to the end of the list.
- cyclic list - a double-linked or single-linked list, in which the last element is also the predecessor of the first. This implementation removes the conditional sequence of items in a list and simplifies operations with them.
- parallel list - A set of several lists. Parallel access to elements of them is possible.

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.



- stack - in one stack elements are added and removed only from one end, observing the LIFO rule (last in first out - from English "the last to enter is the first to leave"), i.e. the most recently added element is the first to access the stack. Thus, operations on the elements are limited.
- queue - access to queue items is also limited as with a stack. However, the FIFO rule (first in, first out - from English "the first to enter is the first to leave") applies here, according to which the element that is in the queue for the longest time (is added the earliest) is processed first. Adding items is done only from the end of the queue, and removing - from the beginning.

Tree-like structures

Tree data structures include different types of trees. [3]

Trees are network structures of data, one of the most important concepts in graph theory. The following are three equivalent definitions of the term "non-oriented tree":

- Connected tree containing n peaks and n-1 edges;
- Connected tree not containing a loop;
- A tree in which each pair of peaks is connected by a simple chain:

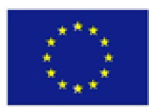
If {\displaystyle G = (X, A)} {\displaystyle G = (X, A)} is a non-oriented tree with {\displaystyle n} n peaks, then any tree formed by its arcs is called a "covering tree", if it includes all the peaks of the tree. Obviously the covering tree has {\displaystyle n-1} {\displaystyle n-1} edges.

Oriented tree: an oriented tree without loops, in which the degree-input of each peak (except one) is equal to 1, and of the marked peak exception (called root) is 0.

Trees are network structures of data, a set of the set X, the elements of which are called peaks, and the set A of ordered pairs of peaks called arcs (edges). The designation is (X, A).

Array

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.



An array is a collection of elements (values or variables) that can be accessed directly through an index.

Comparison of basic data structures

Once we are familiar with the concept of algorithm complexity, we are ready to compare the basic data structures we have considered so far and to assess the complexity of each of them performing basic operations such as adding, searching, deleting and others. This will allow us to easily consider according to the operations we need, which data structure will be most appropriate.

When to use a structure?

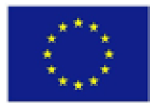
Let's look at each of the data structures listed in the table separately and explain in which situations it is appropriate to use such a structure and how the complexities given in the table are obtained.

Array (T [])

Arrays are ordered sets of a fixed number of elements of a given type (such as numbers) that are accessed by index. Arrays are an area of memory with a specific, predefined size. Adding a new element to an array is a very slow operation, because you actually need to allocate a new array with a dimension larger than 1 of the current one and transfer the old elements to the new array. Array search requires a comparison of each element with the search value. In the average case, N/2 comparisons are required. Deleting from an array is a very slow operation because it involves allocating an array 1 size smaller than the current one and moving all elements without deletion to the new array. Index access is direct and therefore a very fast operation.

Arrays should only be used when we need to process a fixed number of elements that need index access. For example, if we sort numbers, we can write the numbers in an array and apply one of the well-known sorting algorithms. When we need to change the number of elements we work with during operation, the array is not a suitable data structure.

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.



Use arrays when you need to process a fixed number of elements that you need to access by index.

Linked / double linked list (LinkedList <T>)

The linked list and its double linked list variant store an ordered set of elements. Adding is a quick operation, but it's a bit slower than adding to List <T>, because each addition takes up memory. Memory allocation works at a speed that is difficult to predict. Searching in a linked list is a slow operation because it involves crawling all its elements. Accessing an element by index is a slow operation because there is no indexing in the linked list and you have to crawl the list, starting from the first element and moving forward element by element. Deleting an element by index is a slow operation because reaching the element with the specified index is a slow operation. Deleting an element by value is also slow because it involves searching.

The linked list can quickly (with constant complexity) add and delete elements at both ends, making it convenient for implementing stacks, queues, and other similar structures.

Linked list is rarely used in practice because the dynamic-expandable array (List <T>) performs almost all operations that can be performed with LinkedList, but for most of them it works faster and more conveniently.

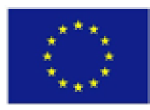
Use List <T> when you need a linked list - it works not slower, but gives you more speed and convenience. Use LinkedList if you need to add and delete elements at both ends of the structure.

Use a linked list (LinkedList <T>) when you need to add and delete elements at both ends of the list. Otherwise, use List <T>.

Dynamic array (List <T>)

The dynamic array (List <T>) is one of the most used data structures in practice. It does not have a fixed size, like arrays, and allows direct access by index, unlike the linked list (LinkedList <T>). The dynamic array is also known as the "array list" and "dynamically expandable array" names.

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.



List <T> internally stores its elements in an array that is larger than the number of stored elements. When adding an element, there is usually free space in the inner array, so this operation takes a constant amount of time. Sometimes the array overflows and needs to be expanded. This takes linear time, but is very rare. Finally, for a large number of additions, the average complexity of adding an element to List <T> is constant - O (1). This average complexity is called amortized complexity. Amortized linear complexity means that if we add 10,000 elements sequentially, we will perform a total number of steps of the order of 10,000 and most of them will be executed in constant time, and the rest (a very small part) will be executed in linear time.

Searching in List <T> is a slow operation because all elements must be crawled. Deletion by index or by value is performed in linear time. Deleting is a slow operation because it involves moving all the elements that are after deleted one position to the left. Index access in List <T> is immediate, for a constant time, as the elements are stored internally in an array.

In practice, List <T> combines the strengths of arrays and lists, making it a preferred data structure in many situations. For example, if we need to process a text file and extract from it all the words that correspond to a regular expression, the most convenient structure in which we can accumulate them is List <T>, because we need a list which length is not previously known and to grow dynamically.

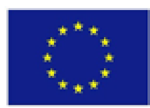
The dynamic array (List <T>) is suitable when we need to add elements often and we want to keep the order of their addition and access them often by index. If we frequently search for or delete an element, List <T> is not an appropriate structure.

Use List <T> when you need to quickly add elements and access them by index.

Stack

A stack is a data structure in which 3 operations are defined: adding an element to the top of the stack, deleting an element from the top of the stack, and retrieving an element from the top of the stack without removing it. All these operations are performed quickly, with constant complexity. Index search and access operations are not supported.

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.



The stack is a structure with LIFO behavior (last in, first out) - last entered, first out. It is used when we need to model such behavior, for example, if we need to keep the path to the current position in recursive search.

Use a stack when you need to implement the last-in-first-out (LIFO) behavior.

Queue

A queue is a data structure in which two operations are defined: adding an element and retrieving the element that is OK. These two operations are performed quickly, with constant complexity, as the queue is usually implemented through a linked list. Remember that the linked list can quickly add and delete elements at both ends.

The behavior of the queue structure is FIFO (first in, first out) - first entered, first out. Index search and access operations are not supported. The queue naturally models a list of waiting people, tasks, or other objects that need to be processed sequentially, in the order in which they enter.

As an example of using a queue we can mention the implementation of the algorithm "search in width", which starts with a given initial element and its neighbors are added to the queue, then processed in their order of receipt, and during their processing their neighbors are added to the queue. This is repeated until the element we are looking for is reached.

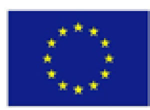
Use a queue when you need to implement the first-in, first-out (FIFO) behavior.

Dictionary implemented with a hash table (Dictionary <K, T>)

The "dictionary" structure assumes storage of key-value pairs and provides fast key search. When implementing with a hash table (Dictionary <K, T> class) in the .NET Framework), adding, searching, and deleting elements work very quickly - with constant complexity in the average case. The index access operation is not available because the elements in the hash table are arranged almost randomly and their order of entry is not preserved.

Dictionary <K, T> stores its elements internally in an array, placing each element in a position given by the hash function. In this way, the array is partially occupied - in some cells

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.



there is a value, while others are empty. If several values need to be placed in the same cell, they are arranged in a linked list (chaining). This is one way to solve the problem of collisions. When the occupancy rate of the hash table exceeds 100% (this is the default value of the load factor parameter), its size doubles and all elements occupy new positions. This operation works with linear complexity, but is performed so rare that the amortized complexity of the add operation remains constant.

The hash table has one feature: with an unfavorably chosen hash function that causes many collisions, basic operations can become quite inefficient and reach linear complexity. In practice, however, this hardly happens. Therefore, the hash table is considered to be the fastest data structure that provides key addition and search.

The hash table in the .NET Framework assumes that each key occurs in it at most once. If we write two elements in succession with the same key, the last one will replace the previous one and in the end we will lose one element. This is an important feature that we must take into account.

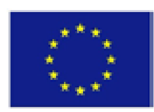
Sometimes we need to store several values in one key. This is not supported by default, but we can use List<T> as a value for this key and accumulate a series of elements in it. For example, if we need a Dictionary <int, string> hash table in which to accumulate pairs {integer, string} with iterations, we can use Dictionary <int, List <string>>.

A hash table is recommended to be used whenever we need a quick key search. For example, if we need to count how many times each word occurs in a text file among a given number of words, we can use Dictionary <string, int> using the search words as a key, and for value - how many times they occur in the file.

Use a hash table when you want to quickly add elements and search by key.

Many programmers (especially beginners) live with the delusion that the main advantage of a hash table is the convenience of looking for a value by its key. In fact, the main advantage is not this at all. We can perform a key search with an array and a list and even a stack. No problem, anyone can implement them. We can define an Entry class that stores a key and a value and work with an array or list of Entry elements. You can perform a search, but in any

This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.



case it will work slowly. This is the big problem with lists and arrays - they don't offer quick search. In contrast, the hash table can search quickly and add new items quickly.

The main advantage of the hash table over other data structures is the extremely quick search and addition of elements. Convenience of work is a secondary factor.

Dictionary implemented with a tree (SortedDictionary <K, T>)

The implementation of the "dictionary" data structure through a red-black tree (class SortedDictionary <K, T>) is a structure that allows storage of key-value pairs, in which the keys are arranged (sorted) by size. The structure provides quick execution of basic operations (adding an element, searching by key and deleting an element). The complexity with which these operations are performed is logarithmic - O (log (N)). This means 10 steps for 1,000 elements and 20 steps for 1,000,000 elements.

In contrast to hash tables, where a poor hash function can lead to a linear complexity of search and addition, in the structure SortedDictionary <K, T> the number of steps to perform basic operations in the average and in the worst case is the same - log2 (N). Balanced trees have no hashing, no collisions and no risk of using a bad hash function.

Again, as with hash tables, a key can occur in the structure at most once. If we want to put several values under the same key, we have to use a list as a value for the elements, for example List<T>.

SortedDictionary <K, T> keeps its elements internally in red-black balanced tree, arranged by key. This means that if we crawl the structure (through its iterator or through a foreach loop in C #), we will get the elements sorted in ascending order by their key. Sometimes this can be very helpful.

Use SortedDictionary <K, T> in cases where a structure is needed in which you can quickly add, search quickly, and need to retrieve elements sorted in ascending order. In general, Dictionary <K, T> works a little faster than SortedDictionary <K, T> and is preferable.

As an example of using SortedDictionary <K, T> we can give the following task: to find all the words in a text file that occur exactly 10 times and to print in alphabetical order. This is a

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

task that we can also solve successfully with Dictionary <K, T>, but we will have to do one more sorting. In solving this problem we can use SortedDictionary <string, int> and go through all the words in the text file and for each of them save in the sorted dictionary how many times it occurs in the file. Then we can go through all the elements of the dictionary and print those of them in which the number of matching is exactly 10. They will be arranged alphabetically, as in the natural internal order of the sorted dictionary.

Use SortedDictionary <K, T> when you want to quickly add elements and search by key and you will then need the elements sorted by key.

Set implemented with a hash table (HashSet <T>)

The data "set" structure is a set of elements, among which there are no repeating ones. The main operations are adding an element to the set, checking for an element to belong to the set (search) and removing an element from the set (deleting). The index search operation is not supported, i.e. we do not have direct access to the elements by order number, because there are no order numbers in this structure.

A set implemented by a hash table (class HashSet <T>) is a special case of a hash table in which we have only keys, and the values stored under each key are irrelevant. This class has only been included in the .NET Framework since version 3.5.

As with the hash table, the main operations in the HashSet data structure are performed with constant complexity O (1). As with the hash table, an unfavorable hash function can lead to a linear complexity of the basic operations, but in practice this almost does not happen.

As an example of using HashSet<T> we can specify the task of finding all different words in a text file.

Use HashSet<T> when you need to quickly add elements to a set and check if an element is from the set.

Set implemented with a tree (SortedSet <T>)

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

A set implemented with a red-black tree is a special case of SortedDictionary <K, T>, in which the keys and the values coincide.

As with the SortedDictionary <K, T> structure, the basic operations in SortedSet <T> are performed with logarithmic complexity O (log (N)), and this complexity is the same in the average and in the worst case.

As an example of using SortedSet <T> we can specify the task of finding all the different words in a text file and printing them in alphabetical order.

Use SortedSet<T> when you need to quickly add elements to a set and check if an element is in the set, and you will also need the elements sorted in ascending order.

7. Object Oriented Programming (OOP)

What is OOP?

OOP stands for Object Oriented Programming.

Object-oriented programming (OOP) is a paradigm in computer programming, n which a software system is modeled as a set of objects, that interact to each other, unlike the traditional view, in which a program is a list of instructions that a computer executes. Each object is able to receive messages, process data and send messages to other objects.

Object-oriented programming gives more flexibility, making it easier to change programs. It is widely popular in software engineering for large-scale projects. OOP is easier to learn than novice programmers, unlike earlier approaches and methodologies, and the OOP approach is simpler to develop and maintain.

Object-oriented programming uses the following concepts:

- Objects – keep data (fields) and functionality together in separate units in one computer program; objects serve as a basis of modularity and structure in an object-oriented computer program. The objects are a standalone units and should be easy to identify. Modularity allows parts of the program to correspond to individual aspects of the problem.
- Abstraction – The ability of a program to ignore some aspects of the information it works with - the ability to focus on the essential. Each object in the system serves as a model of an abstract "agent" that can perform a job, change its status and report on it, and "communicate" with other objects in the system without revealing how these properties are realized.
- Encapsulation – also called “hiding information”: Makes it impossible for users of an object to change its internal state in an unexpected way; only the internal methods of the object have access to its state.

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

- Polymorphism – Different things or objects may have the same interface or respond to the same (by name) message and respond appropriately depending on the nature or type of the object. This allows many different things to be interchangeable. Thus, a variable in program text can contain different objects during program execution and call different methods at different execution times.
- Inheritance – Organizes and supports polymorphism and encapsulation, allowing to define and create objects that are specialized variants of existing objects. New objects can use (and extend) already defined behavior without having to implement that behavior again.

This is actually a number of ideas, most of which exists for a long time. They are brought together, together with the related terminology, to create a programming methodology. It is said that the ideas behind object-oriented programming together are so strong that they have created paradigm shift in programming.

Their exact definitions vary depending on the point of view. For example, static-type languages often have a slightly different view of object-oriented programming than dynamic-type languages, caused by focusing on the properties of programs when compiling, in the first case, and when executing in the second.

Notes: Abstraction is important, but not unique to object-oriented programming. Reusability is an advantage often attributed to object-oriented programming.

Object-oriented programming is often called „paradigm", rather than a style or type of programming to emphasize the understanding that OOP can change the way a software is developed, by changing the way in which programmers and software engineers think for the software.

The OOP paradigm is not inherently a programming paradigm, but a design paradigm. A system is designed by defining the objects that will exist in it, and the code that actually does the job has no connection to the object or the people who use that object because of the encapsulation.

The challenge in the OOP, therefore, is to design a well-thought-out system of objects.

Procedural programming means writing procedures or functions that perform operations on data, while object-oriented programming is about creating objects that contain both data and functions.

Object-oriented programming has several advantages over procedural programming:

- OOP is faster and easier to implement
- The OOP provides a clear structure for the programs
- OOP allows you to create complete reusable applications with less code and shorter development time

What are classes and objects?

Classes and objects are the two main aspects of object-oriented programming.

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

See the following illustration to see the difference between a class and objects:

CLASS	OBJECTS
fruits	apple banana mango

Another example:

CLASS	OBJECTS
Car	Volvo Audi Toyota

So a class is a template for objects, and an object is an instance of a class.

When individual objects are created, they inherit all the variables and functions of the class.

8. Development of large systems

There are two main steps common to all computer program development, regardless of size or complexity. The first step is analysis, followed by a coding step. This kind of very simple implementation concept is actually all that is required if the effort is small enough and if the final product has to be managed by those who built it - as is usually done with computer programs for internal use. But when it comes to large systems, deployment planning focused only on these steps, however, is doomed to failure.

Specialized software systems are complete end products offering complete service in a specific area. The development of a complex software system requires the incorporation of a number of specialized algorithms in it. Upon reaching a certain level of complexity, we move to the modular development of the system - it is divided into functional modules that can work independently and the development is performed module by module. To reduce the risk of failure, larger projects require more than the basic steps - in particular steps to define the requirements before the analysis step; design step between analysis and coding stages; and a testing step after coding. Also, as anyone familiar with iterative development techniques will tell you, if we develop iteratively, with each iterative upgrade of the results of its predecessor,

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

we can further reduce the risk, because at the end of each iteration we move our main line of development forward, i.e. the more we develop, the less risk we have to face

Many additional steps are needed to develop large systems. The analysis and coding stages are still on the agenda, but they are preceded by two levels of requirements analysis, separated by a program design stage and followed by a testing stage. The arrangement of the steps is based on the following concept: as each step progresses and the design becomes more detailed, there is repetition with the previous and next steps, but rarely with the more distant steps in the sequence.

To address this risk, five steps are used:

First, introduce a "preliminary program design" step between requirements generation and analysis to determine storage, time, and operational constraints, and refine the design by collaborating with the analytical input. In other words, create a based architecture to minimize architecturally significant risks. The key factors for ensuring success are the following:

1. Start the process with the designers
2. Design, definition and distribution of data processing modes
3. Write a review document so that everyone understands the system.

Second, document the design. Software management is impossible without a very high level of documentation. In this way, evidence of completeness is provided, when documented, the design becomes realistic. Downstream processes (development, testing, operations, etc.) require strong design documentation to succeed.

Third, "Do it twice" - this is the development of a functional prototype that simulates the high-risk elements of the system that will be developed; then, after making sure that the high-risk elements are addressed, proceed to develop the real thing - the version that will be delivered to the customer.

Fourth: planning, monitoring and control tests. Testing must be completed by independent testers who have not contributed to the design based on documentation established at earlier stages. The visual code scans for errors in the code. Again, this should be done by someone who is not as close to the code as the developer.

There are different methods for organizing these processes in the development of large systems. One is a traditional Waterfall model, and the other offers more flexible management models and is called Agile.

The **Waterfall model** is one of the earliest methodologies developed for building software products. This model divides software processes into different phases, each of which follows a specific order in software development. These phases are:

- Specification of requirements
- Software design
- Implementation and integration
- Testing (or Validation)
- Deployment (or Installation)

- Support

The processes in the waterfall model are linear and sequential. Each of the stages in the development process begins only when the previous phase is fully completed. In strict compliance with the methodology, return to the previous phase of product modification due to changes in requirements is not allowed.

Flexible methodologies (Agile) for software development is an informal collection of methodologies and techniques for managing software development projects. As the name suggests, the focus of flexible methodologies is the idea that software development is a dynamic process in which long-term planning has limited effectiveness. Flexible methodologies are particularly widely used in product development, where through frequent prototyping, manufacturers have the opportunity to receive feedback from customers and adapt the development to the new requirements.

List of some of the most popular flexible methodologies:

- Scrum
- Extreme programming
- Kanban or Lean
- Crystal
- Dynamic Systems Development Method (DSDM)
- Feature-Driven Development (FDD)

9. Databases for System Developers

The database is a collection of data. The organization and arrangement of data can follow a certain pattern. According to the model, two types of databases can be described - **relational and non-relational**.

The most well-known and used type of database so far is the relational (SQL database). The other type of database that uses a non-relational data model is called NoSQL (non SQL / Not only SQL).

9.1.Relational databases (SQL)

Relational databases, also known as "SQL databases", because of the SQL query language, store data in a pre-structured way - in tables arranged in rows (records) and columns. Relationships (relations) can be created between individual data and tables. Tables and relationships between them form a structure that can be represented as scheme.

9.1.1.Data storage model

The individual units of information are contained in single fields arranged in rows in a table.

ID	post_content	post_title	post_status	post_type
1	<p>Welcome to WordPress›...	Hello Site!	publish	post
2	This is an example page. It's...	Sample Page	draft	page

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

Relational database (MySQL) loaded via HeidiSQL

Additional information about the data is recorded through the columns, in additional fields to the row/record.

9.1.2.Using relational databases (SQL)

SQL databases are widely used - for small amounts of information such as a two-page website, to large web or mobile applications, blogs, on-line stores and more. The most famous ready-made content management systems (CMS) support and use relational databases - WordPress, Joomla, Drupal, Magento and others. However, fewer are those that support NoSQL databases (such as Drupal).

9.1.3. Expansion (scalability) of relational databases (SQL)

Vertically. The database is located on one server. To expand, you can increase the power and resources of this server. It is possible for an SQL database to spread across multiple servers, but implementation is usually complex, resource-intensive, and time-consuming. And since these databases do not offer such functionality in a natural way, further development will be needed so that the various hardware points can imitate the operation of one database, on one server. Additional software development will be needed to manage the logic and distribution of data requests between points, as well as to retrieve and aggregate data from different servers.

9.2. Non-relational databases

NoSQL databases are the common name for various database technologies created for modern applications and the vast amount of information they work with. NoSQL databases solve various SQL constraints for:

- easy scalability on server clusters (horizontal scaling);
- support for different types of data structures;
- use in development with flexible methodologies (agile development).

Some NoSQL databases may not fully comply with the ACID (Atomicity, Consistency, Isolation, Durability) transaction model. The ACID model is a collection of transaction properties that can ensure the integrity, completeness, isolation, and resilience of database transactions.

Some NoSQL databases may also not support join operations used in relational databases. Instead, different approaches are used as substitutes for the processing of related data: several requests instead of one; caching, replicating, de-normalizing and nesting data.

9.2.1. Data storage model

Non-relational databases (NoSQL) do not use schemes and data tables. The first record of the data in the example for the WordPress database and the posts table would look similar in a NoSQL database:

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

```
{ ID: 1,
 title: "Hello Site!",
 content: "<p>Welcome to WordPress...",
 status: "publish",
 type: "post"
}
```


For example, the information in MongoDB NoSQL database is stored in JSON-like "documents". The data (called documents) in the document database can be organized into a collection, which is something like an SQL table. The data model is dynamic and it is possible to add new data fields without requiring redesign of the database scheme/structure.

9.2.2.Using NoSQL databases

NoSQL databases are mainly used for applications that work with huge amounts of information and the need for high-speed availability and constant and automatic expansion. Such applications can be massive web applications and mobile applications that serve millions of users. The emergence of the need for this type of database is attributed to the continuously increasing volumes of data in the digital world and the development of Web 2.0. There are NoSQL paid developments, as well as open source ones such as: MongoDB, CouchDB, Druid, Neo4j Community Edition, Redis, Memcache, etc.

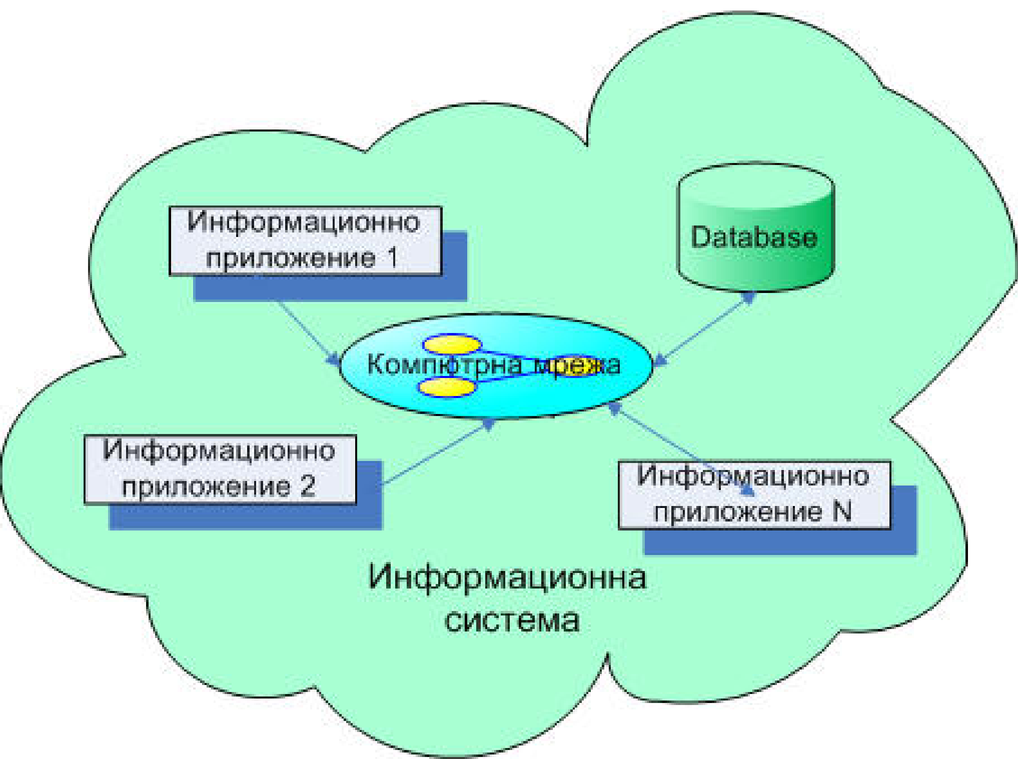
9.2.3.Expansion (scalability)

Horizontally. NoSQL databases have a built-in architecture ability to automatically distribute the database to multiple servers, cloud instances and more. The distributed NoSQL database can handle burdensome requirements for processing and updating large volumes of information in real time, with high operational efficiency.

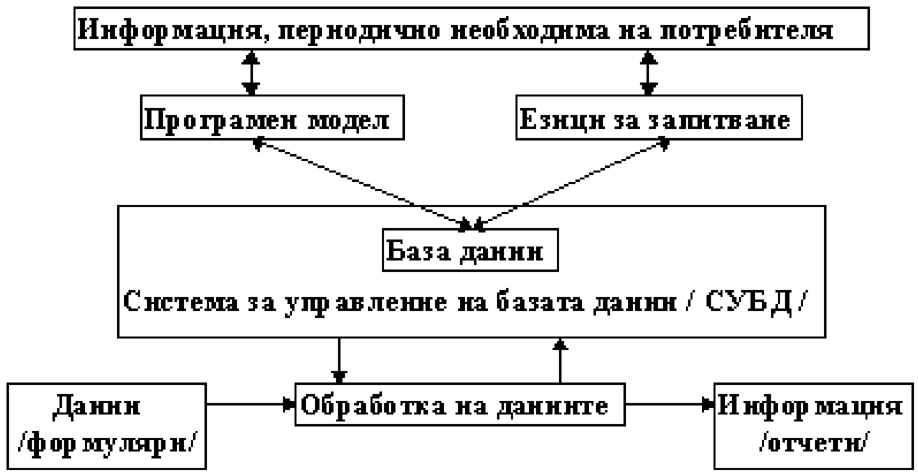
Database management

Data is an important resource of an organization that requires careful planning and management. With the increasing power of computers, it is no longer a problem for each user to create their own database. Today, there are rare cases of information applications that do not use databases.

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.



Database model



Database management system

Complex software packages that support the database and provide an interface with users and user programs.

Main characteristics of DBMS:

- Software that provides users and applications with access to the database.

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

- Presents logical data to users. The details of how and where the data is stored are hidden from users.
- After updating the data, takes care of their compliance.
- Provides and controls the right of access for users.

Database aids

Inquiry languages

They are designed for ordinary users who make queries to the database. They are easy to understand and use and are usually composed of sentences close to spoken English.

Data dictionaries

A set of data about the data itself. Store information about:

- the type of records;
- field names and types;
- other information about the database structure.

Aids for monitoring and evaluation

They determine the degree of use of data by individual users and distribute costs among them.

Report generators

Allow you to set different formats for the output data. They are usually powerful and easy to use.

Types of databases

Hierarchical database model

- The data is stored in a predefined hierarchy;
- The hierarchical tree is turned upside down;
- Quick access;
- Redundant information is stored

Network database model

- The objects from one subject area are united in a network /set/;
- A database consists of several sets, which in turn contain records;
- A set of records may be contained in one or more networks;
- Quick but complex data access, requiring a good knowledge of the logical structure of the data.

Relational database model

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

- The data is stored in tables
- Each table consists of columns /fields/ and rows /records/
- The database is accessed through the links between the individual tables
- Slower access, at the expense of which the work with DB is simplified;
- Provides a relatively high degree of data independence compared to other models.

Object oriented databases

They differ from the traditional relational model and are optimized for object storage. They can store complex types of unstructured data - voice, video, text and images.

10. Development of contract-based systems

Contract design (DBC), also known as contract programming, programming with a contract and custom design programming, is a software design approach.

It instructs software designers to define formal, precise, and verifiable interface specifications for software components that extend the usual definition of abstract data types with preconditions, postconditions, and invariants. These specifications are called "contracts", in accordance with a conceptual metaphor for the terms and obligations of business contracts. The DbC approach assumes that all customer components that call a server component operation will meet the prerequisites specified as required for that operation.

The central idea of DbC is a metaphor for how elements of a software system interact with each other based on mutual obligations and benefits. The metaphor comes from business life, where a "customer" and a "supplier" agree on a "contract" that specifies, for example, that:

- The supplier must provide a specific product (obligation) and has the right to expect that the customer has paid his fee (compensation).
- The customer must pay the fee (obligation) and is entitled to receive the product (benefit).
- Both parties must fulfill certain obligations, such as laws and regulations applicable to all contracts.

Similarly, if the class method in object-oriented programming provides some functionality, it can:

- Expect to be guaranteed a certain login condition by each customer module that calls it: a precondition of the method - an obligation for the customer and a benefit for the provider (the method itself), as it relieves him of the need to handle cases outside the prerequisite.
- Ensure a certain property at the exit: after the condition of the method - an obligation for the provider and an obvious benefit (the main benefit of calling the method) for the customer.
- Maintain a certain property accepted on entry and guaranteed on exit.

Many programming languages have the ability to make claims like these. However, DbC considers these contracts to be so important to the correctness of the software that they should be part of the design process.

The design under the contract also determines criteria for correctness of a software module:

- If the condition for invariant class and is true before the provider is called by the customer, then the invariant and the postcondition will be true after the service is completed.
- When you make calls to a provider, the software module must not violate the provider's prerequisites.

Contract design can also facilitate code reuse, as the contract for each piece of code is fully documented. Module contracts can be considered as a form of software documentation for the behavior of this module. Contract design does not replace regular testing strategies, such as unit testing, integration, and system testing. Rather, it complements external testing with internal self-tests, which can be activated both for isolated tests and in production code during the test phase.

11. System integration

System integration is a broad concept and a more precise definition can hardly be given than the one contained in the name itself - the integration of subsystems for collaboration. Thus, even booting an operating system on a computer can be attributed to this activity, but in practice, as a specialization of companies' system integrators, the term began to be used with the advent of personal computer networks and client-server structure. With the complication of information technologies and with the expansion of their use by various businesses, government organizations, sectors such as medicine and education, research institutions, etc. to the system integration are added more and more complex activities and requirements to the professionalism and capabilities of the companies working in this field. Specializations are even beginning to emerge, such as a telecommunications equipment integrator, a wireless network integrator, but the trend is more towards adding new activities and enlarging companies already operating in the sector.

From a technological point of view, it may be appropriate to say that system integrators must develop an infrastructure from information and telecommunication technologies to the specific applications for a given organization, from where the activity of companies in the application software sector begins. Or system integrators must put into operation all hardware devices with the corresponding system software and middleware, all network and communication functions, including nowadays the use of the Internet, e-mail, mobile devices, wireless networks, etc.

This area of integration is used for the purposes of this catalog, and in a broader sense the term is also used to integrate applications such as ERP, CRM, business process modeling, as well as in many other industries outside of information and telecommunications technology.

Here are some more definitions:

- System integration is the integration of subsystems-components into one system and ensuring their operation as one system.
- System integration is the definition of the "glue" between the individual components.
- System integration is adding value to the system due to the interaction between subsystems.

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

- System integration is ensuring the operation of all parts as a whole.
- System integration is providing customers with what they want to get.

It is most appropriate to say that it is all this. The main steps generally include:

- receiving information from the customer
- project management and terms of reference management
- formulation of requirements
- documenting the requirements
- development of subsystems
- preliminary tests
- full integration
- official tests
- official certification
- acceptance by the customer

In today's connected world, the role of system integration companies as well as engineers in this field is becoming increasingly important - more and more systems are being designed and developed that need to be connected together.

Systems integration engineers need to have a wide range of skills that can hardly be enumerated - they need to know software and hardware engineering, interface protocols, skills to deal with common problems, and so on. It is sometimes said that a system integrator needs to know a little about a wide range of products, but the most important thing is to have the ability to quickly examine available products and be a good diagnostician.

When developing an integration strategy, follow these six basic steps:

STEP 1: Find out who your integration center of gravity is

One of the first questions to consider when planning a hybrid integration in your organization is where you will host (deploy) your integration tools and technology. The answer to this question lies in your "center of gravity".

This means that you must first consider the location of the systems you currently have, including strategic systems for recording and storing information, such as ERP and CRM. Next, consider how you plan to expand your applications and projects over the next three to five years to determine the speed and complexity of moving to the cloud.

Hosting your integrations close to your applications and other data sources is important to avoid slowing down the connection processes between applications. The closer the integration is to your applications, the better performance you can expect from it. You can choose to host your integration locally, in the cloud, or both.

Only in cloud: If your organization doesn't have a lot of legacy investment in this area and has taken a firm stand for the cloud, you may find that a solution entirely based on the cloud is best for you. However, this is not the typical scenario for most organizations that have accumulated a rich IT infrastructure over the years.

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

Locally: For many, the most practical location for integration will continue to be the local one, co-located with the recording and storage systems. This is especially true if your organization currently has only a few cloud-based applications and does not plan to increase their number in the near future. This choice may be most appropriate if you cannot easily move applications to the cloud due to existing regulations or for security reasons.

Hybrid from a cloud and local hosting: For companies that have multiple local applications and legacy investments, but are currently deploying new applications and infrastructure mainly through the cloud, the choice - by default - tends towards a hybrid integration environment with some form of local integration solution, along with a cloud-hosted integration service; for example, an integration Platform-as-a-Service (iPaaS). For most such companies, planning for a cloud-based integration solution should be a priority.

Once your company decides where the integration technology will be hosted, the next question to decide is how much control you want to have over hosting and system management.

STEP 2: Decide how much control and responsibility you want to have

In today's IT environment, companies have much more flexibility than in previous years. One of the areas where this is particularly true is how much practical control and responsibility they choose to exercise over their systems. We are witnessing an increase in the mix of self-managed, exported and hosted systems.

The answer to the question of control and responsibility is often influenced, first of all, by the motivation of your company to implement new cloud services and ranges from full control over the entire installation, development and operations, to complete outsourcing of this responsibility to external service providers. Below you will find the three most common approaches to hybrid integration:

Approach to hybrid integration	**Description**	**Considerations**
Private/public cloud integration	Many organizations are expanding the use of internal private cloud technologies, as well as publicly available cloud infrastructure platforms such as Amazon EC2®, GoogleCompute Engine™ Service and Microsoft AZURE™, to host their applications and IT projects. In making this change, these companies are also shifting their strategic integration technology to these same	The benefit of this option is that companies can deploy their projects faster, and at the same time gain control over capacity growth as needed. This option gives the most control over integration technologies. However, a major challenge with this approach is that companies need to maintain these projects as if they were located locally, which means they will need to perform maintenance, manage updates, etc. If your company has an aggressive strategy for moving a percentage of its IT

	private cloud environments.	infrastructure to the cloud, then this option may be very appropriate.
Hybrid cloud integration	This approach involves using a cloud-based integration service such as iPaaS to integrate cloud-based applications with local integration technologies, such as the Enterprise Service Bus (ESB). This approach offers constant control over your integrations and reduces your responsibilities for maintenance and updates.	This approach provides additional benefits by upgrading those of the private cloud integration approach, as iPaaS is available to a wider range of users, can be used anywhere (because it is cloud-based), and often provides simplified tools. Also, this approach reduces the cost of redesigning existing integrations such as services, connectivity and transformations. Additional benefits are: accelerated integration and scalability based on the number of transactions passing through the system.
External management of cloud integration	External service-managed cloud integration technology allows the transfer of operations such as installation, hosting, and maintenance of integration technology, thus allowing an external service provider to manage the technology.	The benefit of this approach is that it reduces the amount of practical management and operations required of your organization. It also allows you to continue using the integration technology that you are currently using and in which you have invested, but eliminates the need to manage maintenance, upgrades, and many other day-to-day operations.

Once you have determined how much control you want to maintain over your integration environment, you need to consider who the users of the integration service will be.

STEP 3: Know your customers

One of the trends we have seen in the field of integration in recent years is that organizations are changing their approach to the development of integration. The challenges driving these changes came from the growing number of cloud applications that are often acquired by business departments or departments instead of IT, as well as the need to decentralize responsibility for integration so that business departments can gain more control over their projects. Therefore, you need to answer the question: "Who will do the integration work in the future?"

In most organizations, the answer is a combination of traditional integrators involved, non-integration programmers, and a new class of users, often called "citizen integrators."

Attracted traditional integrators: Whether they are part of the company's Integration Competency Center (ICC) or only involved in an integration project, the traditional integrators involved will continue to play a key role in the organizations' integration projects.

This is because some of these projects are critical to the organization and require specialized knowledge and skills that cannot be delegated to non-experts.

Some companies, for example, will not want to risk the integrity of the data in their information recording systems by outsourcing projects to inexperienced developers. Companies will continue to need experts to develop the design and plan for the integration architecture and to ensure that their data integration plans do not damage the company's existing data structure.

Non-integration programmers: Not all systems are critical. For example, we are seeing growth in the introduction of cloud-based systems by departmental buyers looking for systems that are easy to deploy and commission. In such cases, it is often necessary to synchronize data between information recording systems to ensure that the data is correct in each of the systems. This type of integration projects, although often less complex, sometimes contains confidential information.

In the context of a growing number of integration projects, human resources in this area are declining. Therefore, many departments deal with integration on their own. For example, it is increasingly common for departments to have their own specialized IT teams, often referred to as shadow IT. Many companies want the work to be done by non-specialists, as good ones are quite expensive. The price paid for non-specialized developers is much lower than that of specialized ones. Attracted traditional integrators are proficient in software development approaches and technologies, but typically lack the skills typical of attracted traditional integrators. Skills in the field of integration technologies must be easy to learn if non-specialized developers are to be used.

"Citizen integrators" Finally, a category of users who are increasingly common. We are seeing an increase in the number of business users who are charged with the task of maintaining and managing their cloud-based applications, including the integration of data with them.

These "citizen integrators" are typically not IT developers. Instead, they may be business analysts or department staff responsible for deploying SaaS applications within it. For these users, integration tools should be easy to use, offer a more user-friendly interface, and not require knowledge of more complex architectures and integration concepts.

Transferring integration projects to users, outside the core IT integration team, is one way to get more done with less investment. But what are your other options to deal with the requirements of the project?

STEP 4: Plan how to deal with the requirements of the project

As the number of SaaS applications increases, so does the number of integration projects for which you need to provide resources. Involved integration teams find it difficult to meet this need and sometimes seem to be the "bottleneck" in deploying applications. The lag of high-priority integration projects for the departments is unacceptable for business.

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

Given this trend, you need to consider what approaches you will take to satisfy the growing hunger for integration projects, while not investing additional resources to develop integration resources. Consider the following approaches:

Suggest "self-service" integration:

Given the limitations of IT attracted, another option for thinking about the role of IT in integration is to provide a service provider that allows departments to do the integration work themselves. This approach requires IT to establish common architecture, services, access and tools for integration projects and make them available for use by department staff. With this approach, IT ensures that the company is protected from potential destructive problems that may arise as a result of inexperienced users.

Increase asset reuse:

One of the best ways to deal with the growing demand for integration projects is to limit the amount of "new work" that needs to be done. And the easiest way to achieve this is to use existing integration assets effectively, such as services, connectivity, transformations, and distributions. One danger, both for traditional IT integration projects and for projects to create integration from internal departments, in the implementation of multiple integration solutions, is the creation of a repository of integration assets. Customers who have adopted best practices in service reuse in the past, such as companies with Service-Oriented Architecture (SOA) initiatives, are often well positioned to avoid re-integration. But even if you don't have SOA practice, try to take advantage of all your existing assets, provided they can be used by cloud-based integration tools.

When thinking about how to deal with project demand, you should also consider how to provide flexibility for different types of integration projects.

STEP 5: Use a flexible approach for different types of projects

Does simple and non-critical integration cost as much as complex, critical integration? Do you depend on your integration experts for all your projects? Let's look at two trends that affect how organizations approach integration projects.

Suggest multiple integration modes

Not all integration projects are the same. Some are critical to the organization, and some are urgent and may have a short life.

Therefore, you should consider providing an integration tool for simpler and faster projects. Analysts and industry leaders often call this trend in bi-modal IT application development.

Development mode 1	**Development mode 2**
Traditional, complex integration projects fall into the category of Development Mode 1. These traditional projects require stability and	Alternatively, Mode 2 integration projects require rapid, flexible development and most often do not involve business-critical systems.

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

good planning, testing, management and review of the architecture. In the past, all integration projects have been treated in this way.	Development Mode 2 is for fast and innovative projects that are typically driven by business needs. This type of project is where 'citizen integrators' or non-integration developers can be involved in integration projects.

Accept internal integration APIs

In some cases, it may be acceptable to see APIs as a tool for application integration, giving non-specialists the tools for application integration, but eliminating the need for specialized knowledge and skills. Most IT developers feel comfortable with the API for creating new services and connecting systems. However, this approach may also involve some degree of risk in the absence of proper guidance, leading to the emergence of the type of "spaghetti" integration charts that have plagued IT departments for years.

STEP 6: Consider how you will ensure the integrity of the data for your information recording systems

When introducing new integration options in your company, consider how these changes would affect data quality and project stability.

Changes in who develops the project, as well as in the approaches to creating integrations, can be risky, especially in terms of how your data is protected against compromise caused by developers who do not have the necessary knowledge of your system for recording and storing. data. Only authorized users should have access to the data in your application, and there should be an appropriate level of supervision. Appropriate rules and controls are essential to reduce risk and follow all compliance policies in your organization.

It may make sense, for example, to document your integration processes and best practices in order to avoid risks that may arise in your environment. Your organization will want to reduce the risk associated with the introduction of many new integration projects and the fact that non-integration professionals will manage these projects.

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

PRACTICAL EXERCISES AND TESTS

Thematic cycle 1 "Software testing"

Test questions

Question 1. Which of the following statements is incorrect in the context of performance testing?

1. Response time measurement.
2. Measuring transaction percentages.
3. Recovery test.
4. Simulating many users.

Question 2. Which of the following standards determines the test conditions?

1. ANSI / IEEE 829
2. BS7925-2
3. BS7925-1
4. ISO / IEC 12207
5. ANSI / IEEE 729

Question 3. Which of the following activities does not fall under system testing?

1. Performance, load and stress testing.
2. Usability testing
3. Business process based testing.
4. Top-down integration testing.

Question 4. Which of the following is NOT a black box technique?

1. Separation of equivalence
2. Testing the state of the transition
3. Linear code sequence and skip
4. Syntax testing
5. Limit value analysis

Question 5. Which phase of testing individual software modules are combined together as a group?

1. Module testing
2. Integration testing

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

3. White box testing
4. Software testing

Question 6. Which of the following is NOT a static testing technique?

1. Guessing error
2. Rehearsal
3. Data flow analysis
4. Inspections

Question 7. The verification is:

1. Checking that we are developing the right system
2. Checking that we are developing the system correctly
3. It is performed by an independent test team
4. To make sure it's what the user really wants

Question 8. Beta testing is:

1. Performed by customers on their own site
2. Performed by customers on the site of their software
3. Performed by an independent test team
4. Useful for testing ordered software
5. Performed as early as possible in the life cycle

Question 9. What is NOT true - the black box tester:

1. should be able to understand a document for a functional specification or requirements
2. must be able to understand the source code
3. is highly motivated to detect errors
4. is creative to detect system weaknesses

Question 10. The impact analysis helps to decide:

1. Various tools for performing regression testing
2. Exit criteria
3. How many more test cases need to be written
4. How much regression testing should be done

Question 11. The difference between re-testing and regression testing is:

1. Retesting tests again; regression testing looks for unexpected side effects
2. Retesting looks for unexpected side effects; regression testing repeats these tests
3. Retesting is performed after troubleshooting; regression testing is performed earlier
4. Retesting uses different environments, regression testing uses the same environment
5. Retesting is done by developers, regression testing is done by independent testers

Question 12. Determine the statement that applies in the case of exploratory testing:

1. Execution starts only when the design is finalized
2. Includes simultaneous test design and execution
3. Execution starts only when the design is renewed
4. Execution starts only when the design changes

Question 13. Which of the following is not part of the performance tests?

1. Response time measurement
2. Measuring transaction percentages
3. Recovery test
4. Simulating many users
5. Generating many transactions

Question 14. Which of the following is the main task of test planning?

1. Defining the test approach
2. Preparation of test specifications
3. Evaluation of exit and reporting criteria
4. Measurement and analysis of results

Question 15. In which activity of the main test process is the test environment created?

1. Test implementation and execution.
2. Test planning and control
3. Test analysis and design
4. Evaluation of exit and reporting criteria

Question 16. What is the set of activities that ensure that the software properly performs a particular function.

1. Verification
2. Testing
3. Execution
4. Validation

Question 17. The verification is based on a computer.

1. True
2. False

Question 18. ___________ is done in the debug development phase.

1. Coding
2. Testing
3. Troubleshooting
4. Execution

Question 19. Finding or identifying bugs is known as ________

1. Design
2. Testing
3. Troubleshooting
4. Coding

Question 20. What determines the role of software?

1. System design
2. Design
3. System engineering
4. Implementation

Question 21. What do you call testing of individual components?

1. System testing
2. Unit testing
3. Validation check
4. Testing in a black box

Question 22. A testing strategy that tests the application as a whole is:

1. Collection requirement
2. Verification testing
3. Validation check
4. System testing

Question 23. ________ is tested to ensure that the information enters correctly inside and outside the system.

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

1. modular interface
2. local data structure
3. boundary conditions
4. paths

Question 24. The purpose of the requirement phase is

1. to freeze the requirements
2. to understand the needs of users
3. to determine the scope of testing
4. All of the above

Question 25. Which of the following statements about component testing is not true?

1. Component testing must be performed by development
2. Component testing is also known as insulation or modular testing
3. Component testing should have planned completion criteria
4. Component testing does not include regression testing

Question 26. During which test activity can the highest cost faults be found?

1. Execution
2. Design
3. Planning
4. Verification of performance criteria

Question 27. What percentage of software development costs are taken into account when testing software?

1. 10-20%
2. 40-50%
3. 70-80%
4. 5-10%

Question 28. ________ testing ensures that no new defects are introduced after a change in the code

1. Retesting
2. Regression testing

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

3. Both answer 1 and answer 2
4. None of the above

Question 29. What are the purposes of testing

1. Identification of early defects
2. Gaining confidence
3. Defect prevention
4. All of the above

Question 30. In which model does the testing phase begin after the development phase

- V model
- Agile model
- Prototype model
- Waterfall model

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

Exercise tasks

Task 1. Insert the missing part of the code below to display "Hello World!".

```
int main() {
 [     ] << "Hello World!";
 return 0;
}
```

Task 2. Insert a new line after "Hello World" using a special symbol:

```
int main() {
 cout << "Hello World! [     ]";
 cout << "I am learning C ++";
 return 0;
}
```

Task 3. Comments in C++ are written with special symbols. Insert the missing parts:

```
[     ] This is a single-line comment
[     ] This is a multi-line comment [     ]
```

Task 4. Create a variable named myNum and assign a value of 50 to it.

```
[     ] [     ] = [     ];
```

Task 5. Shows the sum of 5 + 10 using two variables: xand y.

```
[     ] [     ] = [     ];
int y = 10;
```

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

```
cout << x + y;
```

Task 6. Create a variable called z, assign x + y to it and display the result.

```
int x = 5;
int y = 10;
[    ] [    ] = x + y;
cout << [    ];
```

Task 7. Fill in the missing parts to create three variables of the same type using a comma-separated list:

```
[    ] x = 5[    ] y = 6[    ] z = 50;
cout << x + y + z;
```

Task 8. Use the correct keyword to get user input stored in the variable x:

```
int x;
cout << "Type a number: ";
[    ] >> [    ];
```

Task 9. Fill in the missing parts to print the sum of two numbers (which is set by the user):

```
int x, y;
int sum;
cout << "Type a number: ";
[    ] >> [    ];
cout << "Type another number: ";
[    ] >> [    ];
sum = x + y;
```

Task 10. Multiply 10 by 5 and print the result.

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

```
cout << 10 [      ] 5;
```

Task 11. Divide 110 by 5 and print the result.

```
cout << 10 [      ] 5;
```

Task 12. Use the correct operator to increase the value of the variable x by 1.

```
int x = 10;
[      ]x;
```

Task 13. Use the add assignment operator to add a value of 5 to the variable x.

```
int x = 10;
x [      ] 5;
```

Task 14. Use the correct function to print the highest value of x and y.

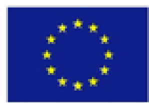
```
int x = 5;
int y = 10;
cout << [      ](x, y);
```

Task 15. Use the correct function to print the square root of x.

```
#include <iostream>
#include <[      ]>
using namespace std;

int main() {
 int x = 64;
 cout << [      ](x);
 return 0;
}
```

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.



Task 16. Use the correct function to round the number 2.6 to its nearest integer.

```
#include <iostream>
#include <[    ]>
using namespace std;

int main() {
 cout << [    ](2.6);
 return 0;
}
```

Task 17. Fill in the missing parts to print the values 1 (for true) and 0 (for false):

```
[    ] isCodingFun = true;
[    ] isFishTasty = false;
cout << [    ];
cout << [    ];
```

Task 18. Fill in the missing parts to print the value true (for true):

```
int x = 10;
int y = 9;
cout << ([    ] [    ] [    ]);
```

Task 19. Add the correct data type for the following variables:

```
[    ] myNum = 9;
[    ] myDoubleNum = 8.99;
[    ] myLetter = 'A';
[    ] myBool = false;
[    ] myText = "Hello World";
```

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

Task 20. Create two Boolean variables called yay and ne and add appropriate values to them:

```
[ ] [ ] = [ ];
[ ] [ ] = [ ];
```

Task 21. Create greeting variable and show its value:

```
[ ] [ ] = "Hello";
cout << [ ];
```

Task 22. Print "Hello World" if x is greater than y.

```
int x = 50;
int y = 10;
[ ] (x [ ] y) {
 cout << "Hello World";
}
```

Task 23. Print "Hello World" if x is equal to y .

```
int x = 50;
int y = 50;
[ ] (x [ ] y) {
 cout << "Hello World";
}
```

Task 24. Print "Yes" if x is equal to y, otherwise print "No".

```
int x = 50;
int y = 50;
[ ] (x [ ] y) {
 cout << "Yes";
} [ ] {
 cout << "No";
}
```

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

Task 25. Print "1" if x is equal to y, print "2" if x is greater than y, otherwise print "3".

```
int x = 50;
int y = 50;
________ (x ________ y) {
 cout << "1";
} ________ (x ________ y) {
 cout << "2";
} ________ {
 cout << "3";
}
```

Task 26. Insert the missing parts to fill in the following "short if...else statement" (terminal operator):

```
int time = 20;
string result = ________time < 18________ ________ "Good day."
________ "Good evening.";
cout << result;
```

Task 27. Insert the missing parts to complete the following switch statement.

```
int day = 2;
switch (________) {
 ________ 1:
 cout << "Saturday";
 break;
 ________ 2:
 cout << "Sunday";
 ________;
}
```

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

Task 28. Fill in the switch statement and add the appropriate keyword at the end to specify some code to run if in the switch expression there is no match by case .

```
int day = 4;
switch ([     ]) {
 [     ] 1:
 cout << "Saturday";
 break;
 [     ] 2:
 cout << "Sunday";
 [     ];
 [     ]:
 cout << "Weekend";
}
```

Task 29. Create a reference variable named meal, which should be a reference to the food variable.

```
string food = "Pizza";
string &[     ] = [     ];
```

Task 30. Get the memory address of the food variable:

```
string food = "Pizza";
cout << &[     ];
```

Task 31. Create a variable of the pointer named ptr, which must point to a stringvariable named food:

```
string food = "Pizza";
[     ] [     ] = &[     ];
```

Task 32. Create a function named myFunction and call it inside main().

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

```
void [      ]() {
 cout << "I just got executed!";
}

int main() {
 [      ];
 return 0;
}
```

Task 33. Insert the missing part to call myFunction twice.

```
void myFunction () {
 cout << "I just got executed!";
}

int main() {
 [      ];
 [      ];
 return 0;
}
```

Task 34. Add fname parameter of type string to myFunction.

```
void myFunction([      ] [      ]) {
 cout << fname << "Doe";
}

int main() {
 myFunction ("John");
 return 0;
}
```

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

Task 35. Insert the missing part to print the number 8 main, като using a specific keyword inside myFunction:

```
int myFunction (int x) {
 [      ] 5 + x;
}

int main() {
 cout << myFunction (3);
 return 0;
}
```

Task 36. Create an array of type string named cars.

```
[      ] [      ][4] = {"Volvo", "BMW", "Ford", "Mazda"};
```

Task 37. Print the value of the second element in the cars array.

```
[      ] [      ][4] = {"Volvo", "BMW", "Ford", "Mazda"};
cout << [      ];
```

Task 38. Change the value from "Volvo" to "Opel" in the cars array.

```
string cars [4] = {"Volvo", "BMW", "Ford", "Mazda"};
[      ] = [      ];
cout << cars [0];
```

Task 39. Check the elements in the cars array.

```
string cars [4] = {"Volvo", "BMW", "Ford", "Mazda"};
[      ] ([      ] = 0; [      ] < 4; [      ]) {
 cout << [      ] << "\n";
}
```

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

Task 40. Fill in the missing part to create a greeting variable of type string and assign it the value Hello.

```
[     ] [     ] = [     ];
```

Task 41. Use the correct operator for connecting of two strings:

```
string firstName = "John";
string lastName = "Doe";
cout << firstName [     ] lastName;
```

Task 42. Use the correct function to print the length of the txt string.

```
string txt = "Hello";
cout << [     ].[     ];
```

Task 43. Access the first character (H) myString and print the result:

```
string myString = "Hello";
cout << [     ];
```

Task 44. Change the first character (H) in myString „J“:

```
string myString = "Hello";
[     ] [     ] [     ];
cout << myString;
```

Task 45. Use the correct function to read a line of text that is placed by the user.

```
string fullName;
cout << "Type your full name: ";
[     ] (cin, fullName);
cout << "Your name is: " << fullName;
```

Task 46. Print I as long as I is less than 6.

```
int i = 1;
```

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

```
[      ] (i < 6) {
 cout << and << "\ n";
 [      ];
}
```

Task 47. Use do/while outline to pring ias long as I is less than 6.

```
int i = 1;
[      ] {
 cout << and << "\ n";
 [      ];
}
[      ] (i <6);
```

Task 48. Use for outline to print "Yes" 5 times:

```
[      ] (int i = 0; i < 5; [      ]) {
 cout << [      ] << "\n";
}
```

Task 49. Stop the outline if i is 5.

```
for (int i = 0; i < 10; i++) {
 if (i == 5) {
 [      ];
 }
 cout << and << "\ n";
}
```

Task 50. In the next loop, when the value is "4", go directly to the next value.

```
for (int i = 0; i < 10; i++) {
 if (i == 4) {
 [      ];
 }
 cout << and << "\ n";
```

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

```
}
```

Test questions

Question 1. The model that demonstrates the implementation of the system is:

1. waterfall model
2. prototype
3. incremental model
4. flexible model

Question 2. Maintenance is the last phase in the waterfall model

1. True
2. False

Question 3. Stage in which the individual components are integrated and it is ensured that they are error-free to meet customer requirements

1. Coding
2. Testing
3. Design
4. Execution

Question 4. ___________ is a step in which the design is translated into machine-readable form.

1. Design
2. Transformation
3. Troubleshooting
4. Coding

Question 5. The customer's requirements are divided into logical modules to facilitate _______________

1. Inheritance
2. Design

3. Editing
4. Implementation

Question 6. What do you call a technical person who is able to understand the basic requirements?

1. Team leader
2. Analyst
3. Engineer
4. Stakeholders

7. A step in the waterfall model, which includes a meeting with the customer to understand the requirements.

1. Collection requirement
2. SRS
3. Execution
4. Customer review

Question 8. Methodology in which the project management processes were step by step.

1. Ascending
2. Waterfall
3. Spiral
4. Prototype

Question 9. An individual who plans and manages the work.

1. Stakeholder
2. Project manager
3. Team leader
4. Programmer

Question 10. Planned program if the work that requires final time, effort and planning to be completed

1. Problem

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

2. Project
3. Process
4. The program

Question 11. Collection of related data is:

1. Information
2. Valuable information
3. Database
4. Metadata

Question 12. DBMS is software.

1. True
2. False

Question 13. The DBMS manages the interaction between __________ and the database.

1. Users
2. Customers
3. End users
4. Stakeholders

Question 14. Which of the following is not included in the DBMS?

1. End users
2. Data
3. Application request
4. HTML

Question 15. The database is usually __________

1. Systematized
2. User oriented
3. Company oriented
4. Data oriented

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

Question 16. A characteristic of a unit is:

1. Attitude
2. Attribute
3. Parameter
4. Restrictions

Question 17. What is IMS?

1. Information learning system
2. Instruction management system
3. Manipulation system for instructions
4. Information management system

Question 18. The model developed by Hammer and Mc Leod in 1981 is:

1. SDM
2. OODBM
3. DDM
4. RDM

Question 19. Object = _________ + relations.

1. data
2. attributes
3. subject
4. restrictions

Question 20. A DBMS is a set of __________ for accessing data

a) Codes

b) Programs

c) Information

d) Metadata

Question 21. DBMS provides a comfortable and efficient environment.

1. True

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

2. False

Question 22. Which of the following is not a level of abstraction?

1. physical
2. logical
3. user level
4. view

Question 23. A level that describes how a record is stored.

1. physical
2. logical
3. user level
4. view

Question 24. The ______ level helps application programs to hide type details

1. physical
2. logical
3. user level
4. view

Question 25. Logical structure of the database.

1. Scheme
2. Attribute
3. Parameter
4. A separate case

Question 26. The actual content in the database at a given time.

1. Scheme
2. Attribute
3. Parameter
4. A separate case

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

Question 27. Which of the following is not an objective logical model?

1. ER
2. Network
3. Semantic
4. Functional view

Question 28. SQL is _______

1. Relational model
2. Network
3. IMS
4. Hierarchical view

Question 29. A level that describes data stored in a database and the relationships between the data.

1. physical
2. logical
3. user level
4. view

Question 30. Each algorithm is a program.

1. True
2. Wrong

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

ANSWERS

Thematic cycle 1 "Software testing"

Answers to test questions

Question 1. Which of the following statements is incorrect in the context of performance testing?

3. Recovery test.

Question 2. Which of the following standards determines the test conditions?

3. BS7925-1

Question 3. Which of the following activities does not fall under system testing?

4. Top-down integration testing.

Question 4. Which of the following is NOT a black box technique?

3. Linear code sequence and skip

Question 5. Which phase of testing individual software modules are combined together as a group?

2. Integration testing

Question 6. Which of the following is NOT a static testing technique?

1. Guessing error

Question 7. The verification is:

2. Checking that we are developing the system correctly

Question 8. Beta testing is:

1. Performed by customers on their own site

Question 9. What is NOT true - the black box tester:

2. must be able to understand the source code

Question 10. The impact analysis helps to decide:

4. How much regression testing should be done

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

Question 11. The difference between re-testing and regression testing is:

1. retesting tests again; regression testing looks for unexpected side effects

Question 12. Determine the statement that applies in the case of exploratory testing:

2. Includes simultaneous test design and execution

Question 13. Which of the following is not part of the performance tests?

3. Recovery test

Question 14. Which of the following is the main task of test planning?

1. Defining the test approach

Question 15. In which activity of the main test process is the test environment created?

1. Test implementation and execution.

Question 16. What is the set of activities that ensure that the software properly performs a particular function.

1. Verification

Question 17. The verification is based on a computer.

1. True

Question 18. ___________ is done in the debug development phase.

1. Coding

Question 19. Finding or identifying bugs is known as ________

2. Testing

Question 20. What determines the role of software? 3. System engineering

Question 21. What do you call testing of individual components?

2. Unit testing

Question 22. A testing strategy that tests the application as a whole is:

4. System testing

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

Question 23. ________ is tested to ensure that the information enters correctly inside and outside the system.

1. modular interface

Question 24. The purpose of the requirement phase is

4. All of the above

Question 25. Which of the following statements about component testing is not true?

Component testing does not include regression testing

Question 26. During which test activity can the highest cost faults be found?

3. Planning

Question 27. What percentage of software development costs are taken into account when testing software?

2. 40-50%

Question 28. ________ testing ensures that no new defects are introduced after a change in the code

2. Regression testing

Question 29. What are the purposes of testing

4. All of the above

Question 30. In which model does the testing phase begin after the development phase

Waterfall model

Thematic cycle 2: "Software development"

Answers to exercise tasks

Task 1. Insert the missing part of the code below to display "Hello World!".

```
int main() {
 [cout] << "Hello World!";
 return 0;
}
```

Task 2. Insert a new line after "Hello World" using a special symbol:

```
int main() {
 cout << "Hello World! [\n] ";
 cout << "I am learning C ++";
 return 0;
}
```

Task 3. Comments in C++ are written with special symbols. Insert the missing parts:

```
[//] This is a single-line comment
[/*] This is a multi-line comment [*/]
```

Task 4. Create a variable named myNum and assign a value of 50 to it.

```
[int] [myNum] = [50];
```

Task 5. Shows the sum of 5 + 10 using two variables: x and y.

```
[int] [x] = [5];
int y = 10;
cout << x + y;
```

Task 6. Create a variable called z, assign x + y to it and display the result.

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

```
int x = 5;
int y = 10;
[int] [z] = x + y;
cout << [z];
```

Task 7. Fill in the missing parts to create three variables of the same type using a comma-separated list:

```
[int] x = 5[,] y = 6[,] z = 50;
cout << x + y + z;
```

Task 8. Use the correct keyword to get user input stored in the x variable:

```
int x;
cout << "Type a number: ";
[cin] >> [x];
```

Task 9. Fill in the missing parts to print the sum of two numbers (which is set by the user):

```
int x, y;
int sum;
cout << "Type a number: ";
[cin] >> [x];
cout << "Type another number: ";
[cin] >> [y];
sum = x + y;
```

Task 10. Multiply 10 by 5 and print the result.

```
cout << 10 [*] 5;
```

Task 11. Divide 10 by 5 and print the result.

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

```
cout << 10 [ / ] 5;
```

Task 12. Use the appropriate operator to increase the value of the variable x by 1.

```
int x = 10;
[ ++ ]x;
```

Task 13. Use the add assignment operator to add a value of 5 to the variable x.

```
int x = 10;
x [ += ] 5;
```

Task 14. Use the correct function to print the highest value of x and y.

```
int x = 5;
int y = 10;
cout << [ max ](x, y);
```

15. Use the correct function to print the square root of x.

```
#include <iostream>
#include <[ cmath ]>
using namespace std;

int main() {
 int x = 64;
 cout << [ sqrt ](x);
 return 0;
}
```

Task 16. Use the correct function to round the number 2.6 to its nearest integer.

```
#include <iostream>
#include <[ cmath ]>
```

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

```
using namespace std;

int main() {
 cout << [round](2.6);
 return 0;
}
```

Task 17. Fill in the missing parts to print the values 1 (for true) and 0 (for false):

```
[bool] isCodingFun = true;
[bool] isFishTasty = false;
cout << [isCodingFu];
cout << [isFishTasty];
```

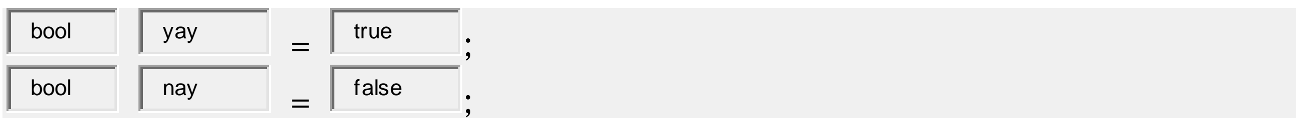
Task 18. Fill in the missing parts to print the value true (for true):

```
int x = 10;
int y = 9;
cout << ([x] [>] [y]);
```

Task 19. Add the correct data type for the following variables:

```
[int]    myNum = 9;
[double] myDoubleNum = 8.99;
[char]   myLetter = 'A';
[bool]   myBool = false;
[string] myText = "Hello World";
```

Task 20. Create two Boolean variables called yay and ne and add to them



The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

Task 21. Create greeting variable and show its value:

```
[string] [greeting] = "Hello";
cout << [greeting];
```

Tasks 22. Print "Hello World" if x is greater than y.

```
int x = 50;
int y = 10;
[if] (x [>] y) {
 cout << "Hello World";
}
```

Task 23. Print "Hello World" if x is equal to y .

```
int x = 50;
int y = 50;
[if] (x [==] y) {
 cout << "Hello World";
}
```

Task 24. Print "Yes" if x is equal to y, otherwise print "No".

```
int x = 50;
int y = 50;
[if] (x [==] y) {
 cout << "Yes";
} [else] {
 cout << "No";
}
```

Task 25. Print "1” if x is equal to y, print "2” if x is greater than y, otherwise print "3".

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

```
int x = 50;
int y = 50;
[if] (x [==] y) {
 cout << "1";
} [else if] (x [>] y) {
 cout << "2";
} [else] {
 cout << "3";
}
```

Task 26. Insert the missing parts to fill in the following "short if...else statement" (terminal operator):

```
int time = 20;
string result = [(]time < 18[)] [?] "Good day."
[:] "Good evening.";
cout << result;
```

Task 27. Insert the missing parts to complete the following switch statement.

```
int day = 2;
switch ([day]) {
 [case] 1:
 cout << "Saturday";
 break;
 [case] 2:
 cout << "Sunday";
 [break];
}
```

Task 28. Fill in the switch statement and add the appropriate keyword at the end to specify some code to run if in the **switch expression there is no match by case .**

```
int day = 4;
```

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

```
switch ( day ) {
 case 1:
 cout << "Saturday";
 break;
 case 2:
 cout << "Sunday";
 break;
 default:
 cout << "Weekend";
}
```

Task 29. Create a reference variable named meal, which should be a reference to the food variable.

```
string food = "Pizza";
string & meal = food;
```

Task 30. Get the memory address of the food variable:

```
string food = "Pizza";
cout << & food;
```

Task 31. Create a variable of pointer named ptr, which must point to a string variable named food:

```
string food = "Pizza";
string* ptr = & food;
```

Task 32. Create a function named myFunction and call it inside main().

```
void myFunction() {
 cout << "I just got executed!";
}
```

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

```
int main() {
 [myFunction];
 return 0;
}
```

Task 33. Insert the missing part to call myFunction twice.

```
void myFunction () {
 cout << "I just got executed!";
}

int main() {
 [myFunction];
 [myFunction];
 return 0;
}
```

Task 34. Add a fname parameter of type string to myFunction.

```
void myFunction([string] [fname]) {
 cout << fname << "Doe";
}

int main() {
 myFunction ("John");
 return 0;
}
```

Task 35. Insert the missing part to print the number 8 main using a specific keyword inside myFunction:

```
int myFunction (int x) {
 [return] 5 + x;
```

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

```
}

int main() {
  cout << myFunction (3);
  return 0;
}
```

Task 36 Create an array of type string named cars.

```
[string] [cars] [4] = {"Volvo", "BMW", "Ford", "Mazda"};
```

Task 37. Print the value of the second element in the cars array.

```
[string] [cars] [4] = {"Volvo", "BMW", "Ford", "Mazda"};
cout << [cars[1]];
```

Task 38. Change the value from "Volvo" to "Opel" in the cars array.

```
string cars [4] = {"Volvo", "BMW", "Ford", "Mazda"};
[cars[0]] = ["Opel"];
cout << cars [0];
```

Task 39. Check the elements in the cars array.

```
string cars [4] = {"Volvo", "BMW", "Ford", "Mazda"};
[for] ([int i] = 0; [i] < 4; [i++]) {
  cout << [cars[i]] << "\n";
}
```

Task 40. Fill in the missing part to create a greeting variable of type string and assign it the value Hello.

```
[string] [greeting] = ["Hello"];
```

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

Task 41. Use the correct operator for connecting of two strings:

```
string firstName = "John";
string lastName = "Doe";
cout << firstName [ + ] lastName;
```

Task 42. Use the correct function to print the length of the txt string.

```
string txt = "Hello";
cout << [ txt ].[ length() ];
```

Task 43. Access the first character (H) myString and print the result:

```
string myString = "Hello";
cout << [ myString[0 ];
```

Task 44. Change the first character (H) in myString „J“:

```
string myString = "Hello";
[ myString ] [ = ] [ 'J' ];
cout << myString;
```

Task 45. Use the correct function to read a line of text that is placed by the user.

```
string fullName;
cout << "Type your full name: ";
[ getline ] (cin, fullName);
cout << "Your name is: " << fullName;
```

Task 46. Print I as long as I is less than 6.

```
int i = 1;
[ while ] (i < 6) {
 cout << and << "\ n";
 [ i++ ];
}
```

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

Task 47. Use do/while outline to pring ias long as I is less than 6.

```
int i = 1;
[do] {
 cout << and << "\ n";
 [i++];
}
[w hile] (i <6);
```

Task 48. Use for outline to print "Yes" 5 times:

```
[for] (int i = 0; i < 5; [i++]) {
 cout << ["Yes"] << "\n";
}
```

Task 49. Stop the outline if i is 5.

```
for (int i = 0; i < 10; i++) {
 if (i == 5) {
 [break];
 }
 cout << and << "\ n";
}
```

Task 50. In the next loop, when the value is "4", go directly to the next value.

```
for (int i = 0; i < 10; i++) {
 if (i == 4) {
 [continue];
 }
 cout << and << "\ n";
}
```

Thematic cycle 2: “Software development”

Answers to test questions

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

Question 1. The model that demonstrates the implementation of the system is:

2. Prototype

Question 2. Maintenance is the last phase in the waterfall model

1. True

Question 3. Stage in which the individual components are integrated and it is ensured that they are error-free to meet customer requirements

2. Testing

Question 4. ___________ is a step in which the design is translated into machine-readable form.

4. Coding

Question 5. The customer's requirements are divided into logical modules to facilitate _______________

4. Implementation

Question 6. What do you call a technical person who is able to understand the basic requirements?

2. analyst

Question 7. A step in the waterfall model, which includes a meeting with the customer to understand the requirements.

1. Collection requirement

Question 8. Methodology in which the project management processes were step by step.

2. Waterfall

Question 9. An individual who plans and manages the work.

2. Project manager

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

Question 10. Planned program if the work that requires final time, effort and planning to be completed

2. Project

Question 11. Collection of related data is:

3. Database

Question 12. DBMS is software.

1. True

Question 13. The DBMS manages the interaction between __________ and the database.

3. End users

Question 14. Which of the following is not included in the DBMS?

4. HTML

Question 15. The database is usually __________

2. User oriented

Question 16. A characteristic of a unit is:

2. Attribute

Question 17. What is IMS?

4. The information management system

Question 18. The model developed by Hammer and Mc Leod in 1981 is:

1. SDM

Question 19. Object = _________ + relations.

3. subject

Question 20. A DBMS is a set of __________ for accessing data

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.

1. Programs

Question 21. DBMS provides a comfortable and efficient environment.

1. True

Question 22. Which of the following is not a level of abstraction?

3. User

Question 23. A level that describes how a record is stored.

1. Physical

Question 24. The ______ level helps application programs to hide type details

4. View

Question 25. Logical structure of the database.

1. Scheme

Question 26. The actual content in the database at a given time.

4. A separate case

Question 27. Which of the following is not an objective logical model?

2. Network

Question 28. SQL is _______

1. Relational model

Question 29. A level that describes data stored in a database and the relationships between the data.

2. Logical

Question 30. Each algorithm is a program.

2. False

The project is co-funded by EU through the Interreg-IPA CBC Bulgaria–Serbia Programme
This publication has been produced with the assistance of the European Union through the Interreg-IPA CBC Bulgaria-Serbia Programme, CCI No 2014TC16I5CB007. The contents of this publication are the sole responsibility of National Association Legal Initiative for Open Government and can in no way be taken to reflect the views of the European Union or the Managing Authority of the Programme.