**The Government of The Russian Federation**

**Federal State Autonomous Educational Institution**

**of Higher Professional Education**

National Research University Higher School of Economics

**Moscow Institute of electronics and mathematics**

**Department of Applied Mathematics**

Syllabus for

**Computer Science and Programming**

**for directions 01.03.04. Applied mathematics bachelor**

**Course Instructors:**

Andrey Vnukov, D.-Eng. (module 1), avnukov@hse.ru

Sergey Aksenov, Ph.D. (module 2-4), [saksenov@hse.ru](mailto:saksenov@hse.ru)

Approved at the meeting of the Department of Applied mathematics "\_\_\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_ 2015 year

Head of Department A. V. Belov \_\_\_\_\_\_\_\_ [signature]

Recommended by Academic Council of educational program

"\_\_\_\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_ 2015 year, No. Protocol\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Approved "\_\_\_" \_\_\_\_\_\_\_\_\_\_\_\_ 2015 year

Academic head of the educational program

Manita L. A. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ [signed]

Moscow, 2015

*This program cannot be used by other divisions of the University and other universities without the permission of the Department-developer of the program.*

1. **Scope and normative references**

This program of academic discipline establishes minimum requirements for knowledge and skills of the student and determines the content and types of studies and reporting.

The program is designed for teachers, leading this discipline, teaching assistants and students in the areas of training 01.03.04."Applied mathematics", studies the discipline of computer science and programming.

The program is designed in accordance with:

* The Federal State Educational Standard (FSES) for the degree of bachelor 01.03.04."Applied mathematics", training - bachelor;
* Educational program 01.03.04."Applied mathematics" bachelor.
* Working curriculum of the University in the direction 01.03.04."Applied mathematics" bachelor approved in 2015.

1. **Goals of learning**

The purpose of learning computer science and programming are:

* **Acquaintance** of students with the discipline of the basics and terminology of low-level programming in Assembler language and object-oriented programming;
* **Acquaintance** of students with the discipline of the basics and terminology of low-level and object-oriented programming. Understanding the capabilities and features of Assembler language and object-oriented programming languages in designing, developing and debugging computer programs;
* **The use** of compatible systems programming. The embed program sections in Assembly language low level program to high level language;
* **Learn** basic principles of low-level and object-oriented programming used in the development and implementation of algorithms, creating and debugging computer programs;
* **The formation** of practical skills of application development in a programming language Assembler and C++.

1. **Competence of the learner, formed as a result of learning the discipline**

As a result of mastering discipline student must:

* **Know** paradigm and programming methodology, features of Assembler language and object-oriented programming languages, the most widely used programming tools; concepts, syntactic and semantic organization, methods of using modern object-oriented programming languages; theoretical and methodological foundations to understand the functionality in the area of information systems development;
* **Be able to apply** in professional activity the Assembler language and modern object-oriented programming languages; to solve problems professionally manufacturing and technological activity taking into account modern achievements of science and technology, including: the development of algorithmic and software solutions in the field of system and applied programming; development of mathematical, information and simulation models on the subject of the performed research;
* **Possess** basic mathematical knowledge and information technologies, to apply them efficiently to solve scientific and technical problems and applied problems associated with the development and use of information technology.
* **To have skills (to gain experience)** formulation of mathematical and information at, cottages, development of technical specifications for software design, applications of mathematical methods to the solution of practical problems, describe algorithms, develop code in languages Assembler and C++, software testing, selection and analysis of materials on the same topic, presentation and protection of the results.

As a result of mastering the discipline a student masters the following competences:

| competence | Code\* in the FSES/ NRU | Descriptors – main features of development (indicators of result achievement) | Forms and methods of training, contributing to the formation and development of competence |
| --- | --- | --- | --- |
| Able to solve problems in professional activity on the basis of analysis and synthesis | UC-3\*\* | Selection and analysis of materials on the same topic. Yourself wrote the program on the instructions of the teacher | Lectures, practical classes, independent work, homework |
| Able to assess the need for resources and plan their use in solving problems in the professional activity | UC-4 | Proficient in the development of algorithmic and software solutions using modern technologies of programming | Lectures, practical classes, independent work, homework |
| Able to work with information: find, evaluate and use information from various sources necessary for the decision of scientific and professional tasks (including on the basis of system approach) | UC-5 | Uses modern standard development environment (IDE) while creating and debugging software products | Lectures, practical classes, independent work, homework |
| Able to conduct research, including problem analysis, setting goals and objectives, the allocation of the object and subject of study, choice of method and research methods, as well as evaluating its quality | UC-6 | Understand the stages and steps of software development. Demonstrates knowledge of modern programming languages of low and high level | Lectures, practical classes, independent work, homework |
| Able to design and develop software components based on modern paradigms, technologies and programming languages | PC-4\*\*\* | Develops the technical specification for the software component using programming languages of low and high level | Lectures, practical classes, independent work, homework |
| Able to apply knowledge of the life cycle of modern projects for the development and operation of software systems and tools for the management of projects in the field of Information technology | PC-8 | Demonstrates knowledge of modern programming languages | Lectures, practical classes, independent work, homework |
| Able to use and develop mathematical modeling techniques and apply analytical and scientific application packages | PC-11 | Uses modern standard development environment (IDE) while creating and debugging software products | Lectures, practical classes, independent work, homework |

\* Federal State Educational Standard (FSES)/ National Research University (NRU);

\*\*(UC-3): Universal Competencies - 3

\*\*\* (PC-4): professional competence - 1.

**Course Description**

**Computer Science and Programming** is a **compulsory** two-semester course for the second year students studying at FAMC specializing in direction of training Applied Mathematics (01.03.04.)

The course aims to provide insights into low-level programming in assembler and object oriented programming basics focusing on C++ application development. The emphasis of the course is on the practical application of the main object oriented programming concepts (Abstraction, Encapsulation, Inheritance, Polymorphism).

**Pre-requisites**

The students are supposed to be familiar with basics of algorithms and computer programming in C.

**Learning Objectives**

The objectives of the Computer Science and Programming course are:

• To acquaint students with low-level and object-oriented programming basics and terminology as well as with capabilities of object-oriented languages;

• To study the language Assembler and object oriented concepts and their implementation in software development;

* Learn basic principles of low-level and object-oriented programming used in the development and implementation of algorithms, creating and debugging computer programs;

• The formation of practical skills of application development in a programming language Assembler and C++.

**Learning Outcomes**

As a result of study, the student should be familiar with object-oriented terminology, object-oriented concepts and the ways of their application in software design.

The student should be able to develop software solutions for the variety of problems using object-oriented concepts, C++ libraries and developer tools for Windows programming.

Students should acquire basic skills and knowledge of object-oriented programming and be able to apply them in practice.

1. **Place of discipline in the structure of the educational program**

Discipline refers to professional cycle (B. 3) and the block of disciplines providing the basic (General professional) training.

The study of this discipline is based on the following disciplines:

* Algorithmic and programming (1th year);
* Computer workshop (1th year);
* Languages and programming techniques;
* Workshop on the computer.

For the development of the discipline, students should posses the following knowledge and competencies:

* To know the basics of programming in language C.
* Know basic methods and algorithms of processing of information structures, methods of analysis of complexity of algorithms;

The main provisions of the discipline should be used further in the study of the following subjects:

The main provisions of the discipline and mastered the competence (UC -3, UC -4, UC -5, UC -6, PC -4, PC -8, PC -11) should be used for further studying of disciplines:

* Operating systems and computer networks (3rd year);
* Numerical methods (3rd year);
* Mathematical modeling (3rd year);
* Interdisciplinary coursework (3rd year);
* Industrial practice (3rd year);
* Programming for the Internet (Web-programming) (4th year);
* Simulation modeling (4 year);
* Numerical methods (4th year);
* Control theory (4th year);
* Computer graphics (4th year);
* Database (data management) (4th year);
* Basics of Web programming.

1. **Thematic plan of the discipline (Distribution of hours)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Topic | Total hours | Contact hours | | | Self-study |
| Lectures | Seminars | Practical Classes |
|  | Generation of programming languages | 7 | 2 |  | 1 | 4 |
|  | The architecture of the microprocessors of the command system of IA-32 and IA-64 | 22 | 8 |  | 4 | 10 |
|  | Programming in Assembler language | 26 | 8 |  | 4 | 16 |
|  | Introduction to C++ programming. | 13 | 4 |  | 1 | 8 |
|  | Introduction to STL. | 18 | 6 |  | 4 | 8 |
|  | User defined data types. | 18 | 6 |  | 4 | 8 |
|  | Inheritance, virtual functions, polymorphisms | 38 | 12 |  | 6 | 20 |
|  | Windows programming basics | 20 | 6 |  | 4 | 10 |
|  | User interphase components | 18 | 6 |  | 2 | 10 |
|  | Document-View architecture | 18 | 6 |  | 2 | 10 |
|  | C++ templates | 14 | 4 |  | 2 | 8 |
|  | Debugging in Visual Studio. Exceptions. | 14 | 4 |  | 2 | 8 |
|  | Multithread programming with MFC | 14 | 4 |  | 2 | 8 |
|  | Total: | **228** | **72** | **0** | **36** | **120** |

1. **Forms of control of knowledge of students**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| The type of control | control form | 2 year | | | | Parameters \*\* |
| 1 | 2 | 3 | 4 |
| Current (week) | homework | 1 |  | 1 | 1 | The creation of the program in the IDE and the protection of its text, 80 minutes |
| intermediate | credit |  | \* |  |  | The creation of the program in the IDE and the protection of its text, 80 minutes |
| final | exam |  |  |  | \* | The creation of the program in the IDE and the protection of its text, 80 minutes |

**6.1.** Evaluation Criteria knowledge and skills

**Homework:** The student must demonstrate the ability to transfer data from one positional numeral system to another and to develop a program in Assembly language on a pre-issued to the job of teacher and defend it in the survey on practical lessons. UC-3, UC-4, UC-5, UC-6, PC-4, PC-8, PC-11.

**Credit:** The student must demonstrate the ability to translate data from one positional number system to another, create computer programs and knowledge of OOP principles within the sections 1 - 3. Competencies: UC-3, UC-4, UC-5, UC-6, PC-4, PC-8, PC-11.

**Exam:** the Student must demonstrate the ability to create computer programs and knowledge of OOP principles within the sections 1 and 13. Competencies: UC-3, UC-4, UC-5, UC-6, PC-4, PC-8, PC-11.

**Homework:** the Student must demonstrate the ability to develop a program on a pre-issued to the job of teacher and defend it in the survey.

**Credit:** the Student must demonstrate the ability to create computer programs in C++ information on topics 4-13. Competencies: UC-3, UC-4, UC-5, UC-6, PC-4, PC-8, PC-11.

**Exam:** the Student must demonstrate the ability to create computer programmes and knowledge of OOP principles within the 4-13. Competencies: UC-3, UC-4, UC-5, UC-6, PC-4, PC-8, PC-11.

Assessment in all forms of current control are set on a 10-point scale.

The inspection should be carried out in a computer class installed with the IDE.

1. **Content of discipline (Course Plan)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Наименование раздела дисциплины | Содержание раздела | Аудиторная работа | Самостоятельная работа | Литература к разделу |
|  | Generation of programming languages | A programming language (program, syntax, semantics). Levels of programming languages (low, intermediate, high). Generation programming languages by the level of abstraction of the specific type of hardware (1 machine, 2 assemblers, 3 procedural, object-oriented, hybrid, domain-specialized, declarative, 4 scripting, descriptive, 5 logical, functional languages). Classification of programming languages depending on the programming paradigm, the degree of abstraction of circuit performance. Common programming languages. The most important programming languages. Pedigree of the most important programming languages and their descendants. | 2 | 10 | [1,3] |
|  | The architecture of the microprocessors of the command system of IA-32 and IA-64 | Structural diagram of a personal computer. The software model of the microprocessor available for programming in the language. The architecture of microprocessors IA-32 (main devices). RISC– and CISC–processors (groups of instruction set). Modified von Neumann architecture. Architecture long command word (VLIW). Registers (General purpose, segment, offset, control and status, debugger, page, flags). The status register processor. Memory structure IA-32. Memory model: Flat Model, Segmented Model, Real-address Mode Model. Common bus (multiplexed, separate, Harvard, super Harvard with the matrix switch). The data transmission channels. The data transfer protocols. Direct access channels to the memory. External device.  The System of instruction set IA-32 (format and a group of instruction set IA-32). Hardware compatibility of the processor architectures of Intel number of "bottom-up" (commands, registers, memory). The emulator commands. Virtual machine.  Compatibility systems programming. The embed program sections in Assembly language in a program in high level language. Description, redundancy and addressing blocks and data.  A group of instruction set. Conditional and unconditional jumps. Cycles. Repeat a string of instruction. Return from CALL. The comparison operation. The mathematical operations. Logical operations. Memory operations. Operations with flags of the processor. Working with BCD numbers.  Principles instruction set of IA-64 (Intel). Comparison instruction set of IA-64 and traditional architectures. The microarchitecture of Intel Itanium 2 processors.  Registers of the x86 architecture.  Instruction set x86-64 architecture. Combining instruction set into groups of instruction set of x86 architecture.  Principles of x86-64 (AMD). Comparison of architectures: x86-64 vs IA-64.  The principle of operation of x86-64. Expansion of registers of the architecture x86-64.  Description of the core representatives of the x86-64 architecture.  Combining instruction set into groups of instruction set architecture x86-64.  Comparison chart Intel and AMD processors 16-, 32-, and 64 - bit architectures. Evolutionary transistor chart showing the architecture. Transistor evolutionary chart "Intel" to compare with the chart "AMD". | 8 | 20 | [1-4] |
|  | Programming in Assembler language | The classical approach to software development. The computing model supported by the Assembler language. Formulation development program in Assembler language: the known model of hardware to develop a local area computing model. General formulation of the software development in Assembler language: the known set of instruction, hardware, local area computing model algorithm to implement the software.  The General structure of program in Assembly language. The syntax of the assembler. The format of instructions and macros.  Operands (permanent/direct, address moved, an address counter, register, base, index, structural, writing).  Operators (arithmetic, shift, comparison, logical, index, override type override segment naming structure type, making segment component of the address expression, the offset of the expression).  Segmentation directives (standard and simplified).  Data types (immediate, simple, complex). Physical and logical interpretation of data. Directive of the description of data of simple types. Attributes of simple data types. Complex data types (arrays, structures, unions, records).  Tools macro Assembler. The macro assembler in the General scheme of compilation of programs. The Macro Instructions. The syntax for macro definitions (in the beginning of the source text in a separate file in the macro library). The context of macro definitions (actual and formal arguments). The process and the result of the substitution arguments (macro generation and macro expansion). Comments. Macro directives (amendments, modification, repetition of lines). Error messages. | 8 | 20 | [4-6] |
|  | Introduction to C++ programming. | Namespaces .  Input/Output in С++.  Increment and decrement operations. Pointers and increment/decrement operations.  References.  Memory allocation in C++.  Constants and macros. | 5 | 8 | [1,2] |
|  | Introduction to STL library | Classes and objects.  Strings and arrays in STL.  STL containers and iterators.  Operations with STL objects.  Type conversation. | 10 | 8 | [1,2] |
|  | User defined data types. | User defined classes.  Using composition in C++.  Initialization of class members (Default constructor, constructor with parameters).  Destructor.  Copy constructor. Ways of passing an object to a function.  Operator overloading.  Static members. Constant members. Class method overloading. | 10 | 8 | [1,4] |
|  | Inheritance, virtual functions, polymorphisms | Access specifiers: public, protected, private.  The concept and mechanisms of inheritance.  Inheritance or composition - Pros and Cons.  Virtual functions and polymorphisms.  Pure virtual functions and abstract classes.  Accessing to an object throughout a pointer.  Type casting and objects.  Indirect base classes.  Virtual destructor.  Multiple inheritance. | 18 | 20 | [1,2] |
|  | Windows programming basics | Function WinMain.  Message loop and message handling.  Smallest Windows application.  Introduction to MFC.  CApp class.  Basic MFC components.  Window classes in MFC.  Basic window messages.  Graphical Device Interphase (GDI). | 10 | 10 | [1] |
|  | User interphase components | Creating window menus.  Creating window toolbars.  Controls (buttons, textboxes, lists, etc.) | 8 | 10 | [1] |
|  | Document-View architecture | SDI and MDI concepts.  Using MS Visual Studio to create SDI and MDI applications.  Creating different views for single document.  View notifications.  Serialization. | 8 | 10 | [1] |
|  | C++ templates | Generic programming basics.  Creating of function templates.  Creating of class templates.  Template parameters.  Inheritance and templates. | 6 | 8 | [1,2] |
|  | Т Debugging in Visual Studio. Exceptions. | Program tracing in Visual Studio.  ASSERT, VERIFY and TRACE macros.  Exceptions. Throwing and catching of exceptions. | 6 | 8 | [1,2] |
|  | Multithread programming with MFC | Creating of working threads.  Creating of interphase threads.  Synchronization.  Terminating threads. | 6 | 8 | [1] |
|  | Total: |  | 108 | 120 |  |

Classes are held in computer class in an interactive form and includes discussions on current tasks and solving problems on the computer.

1. **Educational technology (Methods of Instruction)**

Lectures should be held in classrooms provided with computer and projection equipment.

Practical classes should be held in computer classrooms equipped with the necessary software (i.e. programming) and will include discussions on current tasks and solving problems on the computer.

Students are given different types of tasks.

As a basic template for assignments computer practicum is one option.

Option 1.

A1. a) 4302 b) 79h C1h

A2. ; x = –2h, y = 3h, z = 1h, = –Fh; x = 2Eh, y = 7Fh, z = –FBh, = –8h; x,y,z – byte – word

A3. Given a string of four decimal digits. If the remainder of the division of works of the first and second digit to the fourth digit equals the third digit, replace the third digit of the corresponding capital letter ('0' — 'A', '1', 'B', etc.), otherwise — swap ultra numbers.

A4. Given arrays a and B of 16 bytes. If elements with the same numbers differ by 1 (the number sign), these elements in the B array, replace the elements with the opposite sign (element, replacement of which can cause the overflow, replaced by zero). The address of the modified items to write to the Count array C. the number of such elements.

One of the educational technologies applied in this computer workshop is the incorporation of program sections in Assembly language into a program in a high level language. Translators assembler MASM and TASM and compiler Visual C++ 2010 support segmentation when developing programs.

## Guidelines to the teacher

Not provided.

## Guidance to students

Not provided.

## Methods of Instruction

The course program consists of:

* lectures,
* practical classes,
* regular homework assignment (creating and defending a software solution for a given problem).

1. **Appraisal tools for current control and assessment of student**

**9.1 Subjects of current control tasks**

Sample questions/ tasks for [Insert the name of the current control carried out in written form - control work, Colloquium, homework]:

Not provided.

**9.2. Questions to assess the quality of learning**

The approximate list of questions to offset (exam) throughout the course or to each interim and final monitoring for test students.

1. The syntax, semantics of the programming language.

2. Generation and classification of programming languages.

3. Classification of programming languages (the programming paradigm levels of abstraction, the execution scheme).

4. The programming paradigm levels of abstraction, the execution scheme programming language.

5. Pedigree" most important programming languages.

6. The architecture of microprocessors IA-32 (main devices).

7. RISC– and CISC–processors (groups of instruction set).

8. Registers (General purpose, segment, offset, control and status, debugger, page, flags).

9. Memory model: Flat Model, Segmented Model, Real-address Mode Model.

10. Common bus (multiplexed, separate, Harvard, super Harvard with the matrix switch).

11. Channels and data transfer protocols.

12. The System of instruction set IA-32 (format and a group of instruction set IA-32).

13. Hardware compatibility of the processor architectures of Intel number.

14. Compatibility systems programming.

15. The embed program sections in Assembly language in a program in high level language.

16. Principles of instruction set IA-64 (Intel).

17. Principles of instruction set x86-64 (AMD).

18. Comparison of architectures: x86-64, and IA-64.

19. The principle of operation and expansion of registers of the architecture x86-64.

20. Comparison of Intel and AMD processors 16-, 32-, and 64 - bit architectures.

21. The classical approach to software development. The computing model supported by the Assembler language. Formulation development program in Assembler language.

22. The overall program structure, the syntax of the assembler. The format of instructions and macros.

23. Operands (permanent/direct, address moved, an address counter, register, base, index, structural, writing).

24. Operators (arithmetic, shift, comparison, logical, index, override type override segment naming structure type, making segment component of the address expression, the offset of the expression).

25. Segmentation directives (standard and simplified).

26. Data types (immediate, simple, complex).

27. A direct description of the data.

28. Directive of the description of data of simple types.

29. Complex data types (arrays, structures, unions, records).

30. Tools macro Assembler. The macro assembler in the General scheme of compilation of programs. The Macro Instructions.

31. The syntax of the macro (in the beginning of the source text in a separate file in the macro library).

32. Macro directives (amendments, modification, repetition of lines).

33. There may be bugs.

34. Namespace. Input/output in C++.

35. Operations increment and decrement. Especially operators increment and decrement when working with pointers.

36. Allocation/deallocation in C++.

37. Classes of the standard library for working with strings.

38. Classes of the standard library for containers.

39. Type conversion in C++.

40. The principles of inheritance and composition when designing new classes. Pre-property and disadvantages of inheritance than composition.

41. Initialize data class object (the default Constructor, constructor with parameters). The purpose of the destructor. How to pass an object to a function the copy constructor.

42. Operator overloading.

43. Static variables and methods class. Const methods. The overloaded methods.

44. Virtual functions, concept of polymorphism. Pure virtual functions and abstract classes. Virtual destructor.

45. Multiple inheritance.

46. The concept of generic programming. The class templates. Function templates.

47. Macros ASSERT, VERIFY and TRACE. The concept of exceptions. The mechanism of generation and exception handling.

48. Function WinMain. The message loop. Minimal program for Windows.

49. The architecture of MFC applications. Class CApp. CWindow classes and CFrame.

50. The key messages from the Winwows. The mechanism of message processing in MFC.

51. Drawing in the window, the GDI tools.

52. The main classes of architecture of SDI and how they interact.

53. The main classes of the MDI architecture and principles of their interaction.

54. The concept of serializing, saving, and printing of the document.

55. Create and work with menus. The use of project resources.

56. Create and work with toolbars. The use of project resources.

57. Community work principles with controls (creating, posting, notifications, change).

**9. 3 Examples of tasks the intermediate /final control**

At the request of the author of the program, examples of tickets with questions and tasks, tasks for the test or exam, practice tests on discipline.

1. **Procedure for the formation of estimates on discipline**

(detailed guidelines on formation of estimates on discipline can be given in the Appendix).

The teacher evaluates students ' work at seminars and practical classes: estimated student activity in discussions, speed and accuracy of solving problems. Estimates for work on seminar and practical classes the teacher puts in the work sheet. Accumulated score on a 10-point scale for work on seminar and practical training shall be determined prior to intermediate or final control - Oclassroom.

The teacher assesses the independent work of students: to assess the completeness and correctness of homework, the degree of ownership necessary to perform the job skills. Grades for independent work of the student teacher puts in the work sheet. Accumulated score on a 10-point scale for independent work is defined before intermediate or final control – Ohomework.

The accumulated grade for current control takes into account the results of the student on the current control as follows:

O accumulated= 0.4\* O current + 0.6\* O classroom;

where the current is calculated as a weighted sum of all forms of current control provided for in the Workers educational plans:

O current = O homework;

The resulting score for a discipline is calculated as follows:

O intermediate i = 0.5 O i of the accumulated phase + 0.5 O intermediate test exam

Where i accumulated phase is calculated by the above formula

O cumulative Final= 0.2 O ·intermediate 1+ 0.8 O accumulated 2

The method of rounding cumulative assessment of the intermediate (final) control in the form of the exam: arithmetic.

The sum of weights must be equal to unity: ∑mi = 1, with 0,2 ≤ m1 ≤ 0,8 (according to the Regulation on the organization of knowledge control, the approved FF HSE 24. 06.2011,minutes No. 26).

On the retake the student is not afforded the opportunity to receive additional points for compensation grades for current control.

To offset the student may receive an additional question (an additional practical problem to solve to retake the homework), the answer is scored as 1 point.

On the exam students may request an additional question (an additional practical problem to solve to retake the homework), the answer to which is estimated at 1 point.

Express the views which are taken into account when placing the resulting estimates for intermediate or final control. The sum of weights must be equal to unity: ∑ki = 1, for 0.2 ≤ k1 ≤ 0,8 After all formulas in a mandatory manner is the rounding of the result.

The diploma makes the resultant assessment of academic discipline, which is formed according to the following formula:

O result = 0.4·O cumulative + 0.6 O final

Method of rounding the resultant assessment of academic discipline: arithmetic.

1. **Educational-methodical and information support of discipline (Reading List)**

## 11.1. A basic tutorial

1. Magda S. Y. Assembler for Intel Pentium. –SPb.: Peter, 2006.-410 p., ISBN 5-469-00662-X

## 11.2. The Basic literature (Required)

1. Vnukov A. A. Lectures in electronic form on the course Computer science and programming.– file: Architecture of the instruction set and language Assembler.doc Lectures, 1st module 2015. -145 p.
2. Pirogov V. Y. ASSEMBLER. Training course. - Moscow, Publisher Molgacheva S. V., Publisher of knowledge, 2001. - 848 p., ill., ISBN 5-89251-101-4
3. Golub N. G. The art of programming in Assembler. The lectures and exercises. -2 ed., Rev. and additional –SPb.: LLC "Diasoftyup". 2002.-656 pages ISBN 5-93772-056-35.
4. Rudakov P. I., Finogenov K. G. assembler Language: programming lessons. – DIALOGUE-MEPHI, 2001. -640 p. ISBN 5-86404-160-2
5. Irvine, K. Assembly Language for Intel processors. 3rd edition. Transl. eng. – M:. Publishing house "Williams", 2005 -ISBN: 5-8459-0779-9
6. Vnukov A. A., Raev, A. P., Moshaitin R. V. Organization of the computers, complexes and systems. Tutorial — M.: Moscow state Institute of electronics and mathematics, 2001. — 112 p. ISBN 5-230-16343-7
7. Vnukov A. A. Lectures in electronic form on the course Computer Architecture and systems file: MOSCOW STATE INSTITUTE of ELECTRONICS AND МАТЕМА-ТИКИ -05.doc, lecture, 4th semester 2013. -142 p.
8. Vnukov A. A. Lectures in electronic form on the course Computer Architecture file: Architecture + instruction set 2009.doc, lecture, 4th semester 2009. -128 p.
9. Ivor Horton, Beginning Visual C++ 2010, Indianapolis : Wiley Publishing, Inc., 2011,
10. P.J. Deitel, H.M. Deitel, C++ How to Program, 7th Pearson Education, 2010,

## 11.3. Further reading (Optional)

1. IA-32 Intel Architecture Software Developer’s Manual, Intel® Corp. 2001
2. IA-32 Intel Architecture Optimization, Intel® Corp. 2001
3. Andrew Koenig, Barbara E. Moo, Accelerated C++: Practical Programming by Example // Addison-Wesley, 2001.
4. Scott Meyers, Effective C++, Third Edition, 55 Specific Ways to Improve Your Programs and Designs // Addison-Wesley, 2005.
5. Scott Meyers, More Effective C++ 35 New Ways to Improve Your Programs and Designs // Addison-Wesley, 2005.
6. Scott Meyers, Effective STL // Addison-Wesley, 2005.
7. Herb Sutter, Exceptional C++, 87 New Engineering Puzzles, Programming Problems and Solutions // Addison-Wesley, 2001.
8. Andrei Alexandrescu, Modern C++ Design: Generic Programming and Design Patterns Applied // Addison-Wesley, 2001.

## 11.4. Reference books, dictionaries, encyclopedias

Not provided.

## 11.5. Software

For the successful development of discipline a student uses the following software:

**•** Visual C++ 2010

## 11.6. Remote support of the discipline

Not provided.

1. **Special Equipment and Software Support (if required)**

Lectures should be conducted in a classroom equipped with a projector. For practical training required computer lab equipped with a projector and MS Visual Studio 2010 software.

**Правительство Российской Федерации**

**Федеральное государственное автономное образовательное учреждение высшего профессионального образования   
"Национальный исследовательский университет   
"Высшая школа экономики"**

Московский институт электроники и математики

Департамент Прикладной математики

**Программа дисциплины** Информатика и программирование

для направления 01.03.04. "Прикладная математика" подготовки бакалавра

Авторы программы:

Одобрена на заседании Департамента Прикладной математики «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2015 г.

Руководитель департамента Белов А.В. \_\_\_\_\_\_\_\_ [подпись]

Рекомендована Академическим советом образовательной программы

«\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2015 г., № протокола\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Утверждена «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2015 г.

Академический руководитель образовательной программы

Манита Л.А. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ [подпись]

Москва, 2015

*Настоящая программа не может быть использована другими подразделениями университета и другими вузами без разрешения кафедры-разработчика программы.*

# Область применения и нормативные ссылки

Настоящая программа учебной дисциплины устанавливает минимальные требования к знаниям и умениям студента и определяет содержание и виды учебных занятий и отчетности.

Программа предназначена для преподавателей, ведущих данную дисциплину, учебных ассистентов и студентов направления подготовки 01.03.04."Прикладная математика", изучающих дисциплину Информатика и программирование.

Программа разработана в соответствии с:

* Федеральным Государственным Образовательным Стандартом (ФГОС) по направлению подготовки бакалавра 01.03.04."Прикладная математика", уровень подготовки - бакалавр;
* Образовательной программой 01.03.04."Прикладная математика" подготовки бакалавра.
* Рабочим учебным планом университета по направлению 01.03.04."Прикладная математика" подготовки бакалавра, утвержденным в 2015 г.

# Цели освоения дисциплины

Целями освоения дисциплины Информатика и программирование являются

* **Знакомство** слушателей дисциплины с основами и терминологией низкоуровнего программирования на языке Ассемблер и объектно-ориентированного программирования;
* **Знакомство с**лушателей дисциплины с основами и терминологией низкоуровневого и объектно-ориентированного программирования. Формирование представления о возможностях и особенностях языка Ассемблер и объектно-ориентированных языков программирования при проектировании, разработке и отладке компьютерных программ;
* **Использование** совместимости систем программирования. Встраивание программных секций на языке Ассемблер низкого уровня в программу на языке высокого уровня;
* **Изучение** базовых принципов низкоуровневого и объектно-ориентированного программирования, применяемых при разработке и реализации алгоритмов, создании и отладке компьютерных программ;
* **Формирование** практических навыков разработки прикладных программ на языке программирования Ассемблер и C++.

# Компетенции обучающегося, формируемые в результате освоения дисциплины

В результате освоения дисциплины студент должен:

* **Знать** парадигмы и методологии программирования, особенности языка Ассемблер и объектно-ориентированных языков программирования, наиболее широко используемых языков и средств программирования; концепции, синтаксической и семантической организации, методы использования современных объектно-ориентированных языков программирования; теоретические и методические основы, понимать функциональные возможности в области разработки информационных систем;
* **Уметь** применять в профессиональной деятельности язык Ассемблер и современные объектно-ориентированные языки программирования; профессионально решать задачи производственной и технологической деятельности с учетом современных достижений науки и техники, включая: разработку алгоритмических и программных решений в области системного и прикладного программирования; разработку математических, информационных и имитационных моделей по тематике выполняемых исследований;
* **Владеть** базовыми математическими знаниями и информационными технологиями, эффективно применять их для решения научно-технических задач и прикладных задач, связанных с развитием и использованием информационных технологий;
* **Иметь навыки (приобрести опыт)** постановки математических и информационных задач, разработки технического задания по проектированию программного обеспечения, применения математических методов к решению практических задач, описания алгоритмов решения задачи, разработки программного кода на языках Ассемблер и C++, тестирования программного обеспечения, подбора и анализа материалов по теме задания, презентации и защиты полученных результатов.

В результате освоения дисциплины студент осваивает следующие компетенции:

| Компетенция | Код по ФГОС/ НИУ | Дескрипторы – основные признаки освоения (показатели достижения результата) | Формы и методы обучения, способствующие формированию и развитию компетенции |
| --- | --- | --- | --- |
| Способен решать проблемы в профессиональной деятельности на основе анализа и синтеза | УК-3 | Подбор и анализ материалов по теме задания. Самостоятельно пишет программы по заданию преподавателя | Лекции, практические занятия, самостоятельная работа |
| Способен оценивать потребность в ресурсах и планировать их использование при решении задач в профессиональной деятельности | УК-4 | Владеет навыками разработки алгоритмических и программных решений с использованием современных технологий программирования | Лекции, практические занятия, самостоятельная работа |
| Способен работать с информацией: находить, оценивать и использовать информацию из различных источников, необходимую для решения научных и профессиональных задач (в том числе на основе системного подхода) | УК-5 | Применяет современные стандартные среды разработки (IDE) при создании и отладке программных продуктов | Лекции, практические занятия, самостоятельная работа |
| Способен вести исследовательскую деятельность, включая анализ проблем, постановку целей и задач, выделение объекта и предмета исследования, выбор способа и методов исследования, а также оценку его качества | УК-6 | Понимает стадии и этапы разработки программного обеспечения. Демонстрирует знание современных языков программирования низкого и высокого уровня | Лекции, практические занятия, самостоятельная работа |
| Способен проектировать и разрабатывать компоненты программного обеспечения на основе современных парадигм, технологий и языков программирования | ПК-4 | Разрабатывает техническое задание на разработку программных компонент автоматизированных систем. | Лекции, практические занятия, самостоятельная работа |
| Способен применять знания жизненного цикла современных проектов по созданию и эксплуатации программных систем и инструментальные средства управления проектами в области ИТ. | ПК-8 | Демонстрирует знание современных языков программирования | Лекции, практические занятия, самостоятельная работа |
| Способен использовать и развивать методы математического моделирования и применять аналитические и научные пакеты прикладных программ | ПК-11 | Применяет современные стандартные среды разработки (IDE) при создании и отладке программных продуктов | Лекции, практические занятия, самостоятельная работа |

# Место дисциплины в структуре образовательной программы

Дисциплина относится к профессиональному учебному циклу (Б.3) и блоку дисциплин, обеспечивающих базовую (общепрофессиональную) подготовку.

Изучение данной дисциплины базируется на следующих дисциплинах:

* Алгоритмизация и программирование (1 курс);
* Компьютерный практикум (1 курс);
* Языки и методы программирования;
* Практикум на ЭВМ.

Для освоения учебной дисциплины, студенты должны владеть следующими знаниями и компетенциями:

* Владеть основами программирования на языке С.
* Знать методы и базовые алгоритмы обработки информационных структур, методы анализа сложности алгоритмов;

Основные положения дисциплины и освоенные компетенции (УК-3, УК-4, УК-5, УК-6, ПК-4, ПК-8, ПК-11) должны быть использованы в дальнейшем при изучении дисциплин:

* Операционные системы и сети ЭВМ (3 курс);
* Численные методы (3 курс);
* Математическое моделирование (3курс);
* Междисциплинарная курсовая работа (3курс);
* Производственная практика (3курс);
* Программирование для Интернет (Web-программирование) (4 курс);
* Имитационное моделирование (4 курс);
* Численные методы (4 курс);
* Теория управления (4 курс);
* Компьютерная графика (4 курс);
* Базы данных (управление данными) (4 курс);
* Основы Web программирования.
* Интеллектуальные системы.
* Технологии мультимедиа.

# Тематический план учебной дисциплины

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № | Название раздела | Всего часов | Аудиторные часы | | | Самостоятельная работа |
| Лекции | Семинары | Практические занятия |
|  | Поколения языков программирования | 7 | 2 |  | 1 | 4 |
|  | Архитектура микропроцессоров системы команд IA-32 и IA-64 | 22 | 8 |  | 4 | 10 |
|  | Программирование на языке Ассемблер | 26 | 8 |  | 4 | 16 |
|  | **Введение в программирование на С++.** | 13 | 4 |  | 1 | 8 |
|  | **Знакомство с библиотекой STL.** | 18 | 6 |  | 4 | 8 |
|  | **Построение пользовательских типов данных.** | 18 | 6 |  | 4 | 8 |
|  | **Наследование, виртуальные функции, полиморфизм.** | 38 | 12 |  | 6 | 20 |
|  | **Концепция программирования для Windows. Знакомство с библиотекой MFC.** | 20 | 6 |  | 4 | 10 |
|  | **Элементы пользовательского интерфейса.** | 18 | 6 |  | 2 | 10 |
|  | **Архитектура документ-представление.** | 18 | 6 |  | 2 | 10 |
|  | **Шаблонны С++.** | 14 | 4 |  | 2 | 8 |
|  | **Отладка программ в Visual Studio. Обработка исключений.** | 14 | 4 |  | 2 | 8 |
|  | **Многопоточное программирование в MFC.** | 14 | 4 |  | 2 | 8 |
|  | Всего: | **228** | **72** | **0** | **36** | **120** |

# Формы контроля знаний студентов

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Тип контроля | Форма контроля | 2 год | | | | Параметры \*\* |
| 1 | 2 | 3 | 4 |
| Текущий  (неделя) | Домашнее задание | 1 |  | 1 | 1 | Создание программы в IDE и защита её текста, 80 минут |
| Промежуточный | Зачет |  | \* |  |  | Создание программы в IDE и защита её текста, 80 минут |
| Итоговый | Экзамен |  |  |  | \* | Создание программы в IDE и защита её текста, 80 минут |

## Критерии оценки знаний, навыков

**Домашнее задание**: Студент должен продемонстрировать способность перевода данных из одной позиционной системы счисления в другую и разработать программу на языке Ассемблер по заранее выданному заданию преподавателя и защитить её при опросе на практических занятиях.

**Зачет:** Студент должен продемонстрировать умение создавать компьютерные программы на языке Ассемблер на основе тем разделов 1 - 3. Компетенции: УК-3, УК-4, УК-5, УК-6, ПК-4, ПК-8, ПК-11.

**Экзамен:** Студент должен продемонстрировать умение переводить данные из одной позиционной системы счисления в другую, создавать компьютерные программы и знание принципов ООП в рамках тем разделов 1 - 13. Компетенции: УК-3, УК-4, УК-5, УК-6, ПК-4, ПК-8, ПК-11.

**Домашнее задание**: Студент должен продемонстрировать способность разработать программу по заранее выданному заданию преподавателя и защитить её при опросе.

**Зачет**: Студент должен продемонстрировать умение создавать компьютерные программы на языке С++ основе информации по темам 4-13. Компетенции: УК-3, УК-4, УК-5, УК-6, ПК-4, ПК-8, ПК-11.

**Экзамен:** Студент должен продемонстрировать умение создавать компьютерные программы и знание принципов ООП в рамках тем 4-13. Компетенции: УК-3, УК-4, УК-5, УК-6, ПК-4, ПК-8, ПК-11.

Оценки по всем формам текущего контроля выставляются по 10-ти балльной шкале.

Проведение контроля должно осуществляться в компьютерном классе с установленным IDE.

# Содержание дисциплины

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Наименование раздела дисциплины | Содержание раздела | Аудиторная работа | Самостоятельная работа | Литература к разделу |
|  | Поколения языков программирования | Язык программирования (программа, синтаксис, семантика). Уровни языков программирования (низкий, промежуточный, высокий). Поколения языков программирования по уровню абстрагирования от конкретного типа аппаратной части (1 машинные, 2 ассемблеры, 3 процедурные, объектно-ориентированные, гибридные, доменоспециализированные, декларативные, 4 скриптовые, описательные, 5 логические, функциональные языки). Классификация языков программирования (парадигма программирования, степень абстракции, схема выполнения). Распространенные языки программирования (важнейшие языки программирования, родословная» важнейших языков программирования и их потомков). Причины распространения языка программирования и нераспространения концепций. | 2 | 10 | [1,3] |
|  | Архитектура микропроцессоров системы команд IA-32 и IA-64 | Структурная схема персонального компьютера. Программная модель микропроцессора, доступная для программирования на языке. Архитектура микропроцессоров IA-32 (основные устройства). Архитектура RISC– и CISC–процессоров (группы команд). Модифицированная фон-неймановская архитектура. Архитектура длинного командного слова (VLIW). Регистры (общего назначения, сегментов, офсета, состояния и управления, дебаггера, контроля страниц, флагов). Регистр состояния (флаггов) процессора. Структура памяти IA-32. Модели памяти: сплошная (Flat Model), сегментированная (Segmented Model), режима реального адреса (Real-address Mode Model). Общая шина (мультиплексированная, раздельная, гарвардская, супергарвардская с матричным коммутатором). Каналы передачи данных. Протоколы передачи данных. Каналы прямого доступа в память. Внешние устройства.  Система команд IA-32 (формат и группы команд IA-32). Аппаратная совместимость архитектур процессоров ряда Intel "снизу вверх" (команды, регистры, память). Эмулятор команд. Виртуальная машина.  Совместимость систем программирования. Встраивание программных секций на ассемблере в программу на языке высокого уровня. Поддержка технологии сегментации трансляторами MASM и TASM, компилятором Visual С++ 2010. Описание, резервирование и адресация блоков и данных.  Группы команд. Условные и безусловные переходы. Циклы. Повтор строковых команд. Возврат из CALL. Операции сравнения. Математические операции. Логические операции. Операции с памятью. Операции с флагами процессора. Работа с BCD числами.  Принципы архитектуры IA-64 (Intel). Сравнение IA-64 и традиционных архитектур. Микроархитектура процессоров Intel Itanium 2.  Регистры архитектуры x86.  Команды архитектуры x86-64. Объединение команд в группы команд архитектуры х86.  Принципы архитектуры x86-64 (AMD). Сравнения архитектур: x86-64 против IA-64.  Принцип работы x86-64. Расширение состава регистров архитектуры х86-64.  Описание ядра представителей архитектуры x86-64.Объединение команд в группы команд архитектуры х86-64.  Диаграммы сравнения процессоров Intel и AMD 16-, 32-, 64- битных архитектур. Эволюционно транзисторная диаграмма с указанием разрядности архитектуры. Эволюционно транзисторная диаграмма "компании Intel" для сравнения с диаграммой "компании AMD". | 8 | 20 | [1-4] |
| 1. 2 | Программирование на языке Ассемблер | Классический подход к разработке программного обеспечения. Модель вычислений, поддерживаемая языком Ассемблер. Общая постановка задачи разработки программного обеспечения на языке Ассемблер: по известному набору команд, аппаратного обеспечения, локальной вычислительной модели алгоритма реализовать программное обеспечение.  Общая структура программы на языке ассемблер. Синтаксис ассемблера. Формат команд и макрокоманд.  Операнды ([постоянные/непосредственные,](http://www.kolasc.net.ru/cdo/programmes/assembler/com_1_1.html#ПостоянныеОперанды) [адресные](http://www.kolasc.net.ru/cdo/programmes/assembler/com_1_1.html#АдресныеОперанды), [перемещаемые](http://www.kolasc.net.ru/cdo/programmes/assembler/com_1_1.html#ПеремещаемыеОперанды), [счетчик адреса](http://www.kolasc.net.ru/cdo/programmes/assembler/com_1_1.html#СчетчикАдреса), [регистровый](http://www.kolasc.net.ru/cdo/programmes/assembler/com_1_1.html#РегистровыйОперанд), [базовые, индексные](http://www.kolasc.net.ru/cdo/programmes/assembler/com_1_1.html#БазовыйИиндексныйОперанды), [структурные](http://www.kolasc.net.ru/cdo/programmes/assembler/com_1_1.html#СтруктурныеОперанды), [записи](http://www.kolasc.net.ru/cdo/programmes/assembler/com_1_1.html#Записи)).  Операторы (арифметические, [сдвига](http://www.kolasc.net.ru/cdo/programmes/assembler/com_1_2.html#ОператорыСдвига), [сравнения](http://www.kolasc.net.ru/cdo/programmes/assembler/com_1_2.html#ОператорыСравнения), [логические,](http://www.kolasc.net.ru/cdo/programmes/assembler/com_1_2.html#ЛогическиеОператоры) [индексные,](http://www.kolasc.net.ru/cdo/programmes/assembler/com_1_2.html#ИндексныйОператор) [переопределения типа](http://www.kolasc.net.ru/cdo/programmes/assembler/com_1_2.html#ОператорПереопределенияТипа), [переопределения сегмента](http://www.kolasc.net.ru/cdo/programmes/assembler/com_1_2.html#ОператорПереопределенияСегмента), [именования типа структуры](http://www.kolasc.net.ru/cdo/programmes/assembler/com_1_2.html#ОператорИменованияТипа), [получения сегментной составляющей адреса выражения](http://www.kolasc.net.ru/cdo/programmes/assembler/com_1_2.html#SEG), [получения смещения выражения](http://www.kolasc.net.ru/cdo/programmes/assembler/com_1_2.html#Offset)).  Директивы сегментации (стандартные и упрощенные).  Типы данных (непосредственные, простые, сложные). Физическая и логическая интерпретация данных. Директивы описания данных простых типов. Атрибуты простых типов данных. Сложные типы данных (массивы, структуры, объединения, записи).  Макросредства Ассемблера. Макроассемблер в общей схеме трансляции программы. Макрокоманда. Синтаксис макроопределения (в начале исходного текста, в отдельном файле, в макробиблиотеке). Учет контекста макроопределения (фактические и формальные аргументы). Процесс и результат замещения аргументов (макрогенерация и макрорасширение). Комментарии. Макродирективы (изменения, модификации, следования строк). Сообщения об ошибках. | 8 | 20 | [4-6] |
|  | **Введение в программирование на С++.** | Пространства имен. Ввод/вывод в С++. Операции инкремента и декремента. Особенности операторов инкремента и декремента при работе с указателями. Ссылки. Выделение/освобождение памяти в С++. Константы и макроопределения. | 5 | 8 | [1,2] |
|  | **Знакомство с библиотекой STL.** | Понятие объекта и класса. Классы стандартной библиотеки для работы со строками. Классы стандартной библиотеки для работы с контейнерами. Операции с объектами стандартной библиотеки. Преобразование типов. | 10 | 8 | [1,2] |
|  | **Построение пользовательских типов данных.** | Создание пользовательских классов. Принцип композиции при конструировании новых классов. Инициализация данных объекта класса (Конструктор по умолчанию, конструктор с параметрами). Назначение деструктора. Способы передачи объекта в функцию, конструктор копирования. Перегрузка операторов. Статические переменные и методы класса. Константные методы. Перегруженные методы. | 10 | 8 | [1,4] |
|  | **Наследование, виртуальные функции, полиморфизм.** | Спецификаторы доступа public, protected, private. Понятие и механизм наследования классов. Преимущества и недостатки наследования по сравнению с композицией. Виртуальные функции, понятие полиморфизма. Чистые виртуальные функции и абстрактные классы. Доступ к объекту через указатель, преобразование типов. Непрямые базовые классы. Виртуальный деструктор. Множественное наследование. | 18 | 20 | [1,2] |
|  | **Концепция программирования для Windows. Знакомство с библиотекой MFC.** | Функция WinMain. Цикл обработки сообщений. Минимальная программа для Windows. Использование библиотеки MFC. Класс Capp. Основные элементы программ на базе MFC. Типы классов окон MFC. Основные сообщения Winwows. Графические возможности GDI | 10 | 10 | [1] |
|  | **Элементы пользовательского интерфейса.** | Создание и работа с меню. Создание и работа с панелями инструментов. Поддерживаемые элементы управления (кнопки, текстовые поля, списки и т.д.). Создание, редактирование, получение уведомлений от элементов управления. | 8 | 10 | [1] |
|  | **Архитектура документ-представление.** | Использование мастера Visual Studio для создания SDI и MDI приложений. Архитектура SDI приложений. Архитектура MDI приложений. Различные представления одного документа. Уведомление представлений об изменении документа. Понятие сериализации, сохранение и печать документа. | 8 | 10 | [1] |
|  | **Шаблонны С++.** | Понятие обобщенного программирования. Определение шаблона класса. Создание объектов шаблона класса. Наследование шаблонных классов. Шаблоны функций. | 6 | 8 | [1,2] |
|  | **Отладка программ в Visual Studio. Обработка исключений.** | Трассировка программы в Visual Studio. Макросы ASSERT, VERIFY и TRACE. Понятие исключения. Механизм генерации и обработки исключений. | 6 | 8 | [1,2] |
|  | **Многопоточное программирование в MFC.** | Создание потоков. Рабочие и интерфейсные потоки. Синхронизация потоков. Остановка потоков. | 6 | 8 | [1] |
|  | Всего: |  | 108 | 120 |  |

Занятия проводятся в компьютерном классе в интерактивной форме, включают в себя дискуссии по текущим заданиям и решение задач на ЭВМ.

# Образовательные технологии

Лекции должны проводиться в классах, обеспеченных компьютером и проекционным оборудованием.

Практические занятия должны проводиться в компьютерных классах оснащенных необходимым программным обеспечением (средствами разработки программ) и включать в себя дискуссии по текущим заданиям и решение задач на ЭВМ.

Студентам выдаются разные типовые варианты заданий.

В качестве типового варианта заданий компьютерного практикума приводится один вариант.

Вариант 1.

1. A1. а) 4302 б) 79h C1h

А2. ; x = –2h, y = 3h, z = 1h, = –Fh; x = 2Eh, y = 7Fh, z = –FBh, = –8h; x,y,z – байты, – слово



A3. Дана строка из четырех десятичных цифр. Если остаток от деления произведения первой и второй цифр на четвертую цифру равен третьей цифре, то заменить третью цифру соответствующей прописной буквой ('0' — на 'A', '1' — на 'B' и т.д.)., иначе — поменять местами крайние цифры.

A4. Даны массивы A и B из 16 байтов. Если элементы с одинаковыми номерами отличаются на 1 (числа знаковые), то такие элементы в массиве B заменить элементами с противоположным знаком (элемент, замена которого может вызвать переполнение, заменить нулем). Адреса изменённых элементов записать в массив C. Сосчитать количество таких элементов.

Одной из образовательных технологий, применяемых в данном компьютерном практикуме, является встраивание программных секций на ассемблере в программу на языке высокого уровня. Трансляторы с ассемблера MASM и TASM, а также компилятор Visual С++ 2010 поддерживают технологию сегментации при разработке программ.

## Методические рекомендации преподавателю

Не предусмотрены.

## Методические указания студентам

Не предусмотрены.

## Методы обучения

Программа курса состоит из:

• лекций,

• практических занятий,

• регулярных домашних заданий (составление и защита программного решения для данной проблемы).

# Оценочные средства для текущего контроля и аттестации студента

## 9.1 Тематика заданий текущего контроля

Примерные вопросы/ задания для :

Не предусмотрены.

## 9.2 Вопросы для оценки качества освоения дисциплины

Примерный перечень вопросов к зачету (экзамену) по всему курсу или к каждому промежуточному и итоговому контролю для самопроверки студентов.

1. Синтаксис, семантика языка программирования.
2. Поколения и классификация языков программирования.
3. Классификация языков программирования (парадигма программирования, степень абстракции, схема выполнения).
4. Парадигма программирования, степень абстракции, схема выполнения языка программирования.
5. Родословная» важнейших языков программирования.
6. Архитектура микропроцессоров IA-32 (основные устройства).
7. Архитектура RISC– и CISC–процессоров (группы команд).
8. Регистры (общего назначения, сегментов, офсета, состояния и управления, дебаггера, контроля страниц, флагов).
9. Модели памяти: плошная (Flat Model), сегментированная (Segmented Model), режима реального адреса (Real-address Mode Model).
10. Общая шина (мультиплексированная, раздельная, гарвардская, супергарвардская с матричным коммутатором).
11. Каналы и протоколы передачи данных.
12. Система команд IA-32 (формат и группы команд IA-32).
13. Аппаратная совместимость архитектур процессоров ряда Intel.
14. Совместимость систем программирования.
15. Встраивание программных секций на ассемблере в программу на языке высокого уровня.
16. Принципы архитектуры IA-64 (Intel).
17. Принципы архитектуры x86-64 (AMD).
18. Сравнения архитектур: x86-64 и IA-64.
19. Принцип работы и расширение состава регистров архитектуры х86-64.
20. Сравнения процессоров Intel и AMD 16-, 32-, 64- битных архитектур.
21. Классический подход к разработке программного обеспечения. Модель вычислений, поддерживаемая языком Ассемблер. Постановка задачи разработки программы на языке Ассемблер.
22. Общая структура программы, синтаксис ассемблера, формат команд и макрокоманд.
23. Операнды ([постоянные/непосредственные,](http://www.kolasc.net.ru/cdo/programmes/assembler/com_1_1.html#ПостоянныеОперанды) [адресные](http://www.kolasc.net.ru/cdo/programmes/assembler/com_1_1.html#АдресныеОперанды), [перемещаемые](http://www.kolasc.net.ru/cdo/programmes/assembler/com_1_1.html#ПеремещаемыеОперанды), [счетчик адреса](http://www.kolasc.net.ru/cdo/programmes/assembler/com_1_1.html#СчетчикАдреса), [регистровый](http://www.kolasc.net.ru/cdo/programmes/assembler/com_1_1.html#РегистровыйОперанд), [базовые, индексные](http://www.kolasc.net.ru/cdo/programmes/assembler/com_1_1.html#БазовыйИиндексныйОперанды), [структурные](http://www.kolasc.net.ru/cdo/programmes/assembler/com_1_1.html#СтруктурныеОперанды), [записи](http://www.kolasc.net.ru/cdo/programmes/assembler/com_1_1.html#Записи)).
24. Операторы (арифметические, [сдвига](http://www.kolasc.net.ru/cdo/programmes/assembler/com_1_2.html#ОператорыСдвига), [сравнения](http://www.kolasc.net.ru/cdo/programmes/assembler/com_1_2.html#ОператорыСравнения), [логические,](http://www.kolasc.net.ru/cdo/programmes/assembler/com_1_2.html#ЛогическиеОператоры) [индексные,](http://www.kolasc.net.ru/cdo/programmes/assembler/com_1_2.html#ИндексныйОператор) [переопределения типа](http://www.kolasc.net.ru/cdo/programmes/assembler/com_1_2.html#ОператорПереопределенияТипа), [переопределения сегмента](http://www.kolasc.net.ru/cdo/programmes/assembler/com_1_2.html#ОператорПереопределенияСегмента), [именования типа структуры](http://www.kolasc.net.ru/cdo/programmes/assembler/com_1_2.html#ОператорИменованияТипа), [получения сегментной составляющей адреса выражения](http://www.kolasc.net.ru/cdo/programmes/assembler/com_1_2.html#SEG), [получения смещения выражения](http://www.kolasc.net.ru/cdo/programmes/assembler/com_1_2.html#Offset)).
25. Директивы сегментации (стандартные и упрощенные).
26. Типы данных (непосредственные, простые, сложные).
27. Непосредственное описание данных.
28. Директивы описания данных простых типов.
29. Сложные типы данных (массивы, структуры, объединения, записи).
30. Макросредства Ассемблера. Макроассемблер в общей схеме трансляции программы. Макрокоманда.
31. Синтаксис макроопределения (в начале исходного текста, в отдельном файле, в макробиблиотеке).
32. Макродирективы (изменения, модификации, следования строк).
33. Возможные ошибки.
34. Пространства имен. Ввод/вывод в С++.
35. Операции инкремента и декремента. Особенности операторов инкремента и декремента при работе с указателями.
36. Выделение/освобождение памяти в С++.
37. Классы стандартной библиотеки для работы со строками.
38. Классы стандартной библиотеки для работы с контейнерами.
39. Преобразование типов в C++.
40. Принципы наследования и композиции при конструировании новых классов. Преимущества и недостатки наследования по сравнению с композицией.
41. Инициализация данных объекта класса (Конструктор по умолчанию, конструктор с параметрами). Назначение деструктора. Способы передачи объекта в функцию, конструктор копирования.
42. Перегрузка операторов.
43. Статические переменные и методы класса. Константные методы. Перегруженные методы.
44. Виртуальные функции, понятие полиморфизма. Чистые виртуальные функции и абстрактные классы. Виртуальный деструктор.
45. Множественное наследование.
46. Понятие обобщенного программирования. Шаблоны классов. Шаблоны функций.
47. Макросы ASSERT, VERIFY и TRACE. Понятие исключения. Механизм генерации и обработки исключений.
48. Функция WinMain. Цикл обработки сообщений. Минимальная программа для Windows.
49. Архитектура приложений MFC. Класс CApp. Классы CWindow и CFrame.
50. Основные сообщения Winwows. Механизм обработки сообщений в MFC.
51. Рисование в окне, инструменты GDI.
52. Основные классы архитектуры SDI и принцип их взаимодействия.
53. Основные классы архитектуры MDI и принципы их взаимодействия.
54. Понятие сериализации, сохранение и печать документа.
55. Создание и работа с меню. Использование ресурсов проекта.
56. Создание и работа с панелями инструментов. Использование ресурсов проекта.
57. Общине принципы работы с элементами управления (создание, размещение, получение уведомлений, изменение).

## 9. 3 Примеры заданий промежуточного /итогового контроля

По желанию автора программы, приводятся примеры билетов с вопросами и задачами, заданий для зачета или экзамена, тренировочные тесты по дисциплине.

Не предусмотрены при выполнении компьютерного практикума по языку программирования Ассемблер.

# Порядок формирования оценок по дисциплине

(подробные методические рекомендации по формированию оценок по дисциплине могут быть приведены в приложении)

Преподаватель оценивает работу студентов на семинарах и практических занятиях: оценивается активность студента в дискуссиях, скорость и правильность решения задач. Оценки за работу на семинарских и практических занятиях преподаватель выставляет в рабочую ведомость. Накопленная оценка по 10-ти балльной шкале за работу на семинарских и практических занятиях определяется перед промежуточным или итоговым контролем - *Оаудиторная*.

Преподаватель оценивает самостоятельную работу студентов: оценивается полнота и правильность выполнения домашних работ, степень владения необходимыми для выполнения работы навыками. Оценки за самостоятельную работу студента преподаватель выставляет в рабочую ведомость. Накопленная оценка по 10-ти балльной шкале за самостоятельную работу определяется перед промежуточным или итоговым контролем – *Од/з*.

Накопленная оценка за текущий контроль учитывает результаты студента по текущему контролю следующим образом:

О*накопленная*= 0.4\* *Отекущий* + 0.6\* *Оаудиторная*;

где*Отекущий* рассчитывается как взвешенная сумма всех форм текущего контроля, предусмотренных в РУП:

*Отекущий* = *Од/з*;

Результирующая оценка за дисциплину рассчитывается следующим образом:

*Опромежуточная i* = 0.5*·О**накопленная i этапа +* 0.5·*Опромежуточный зачет/экзамен*

Где *Онакопленная i этапа* рассчитывается по приведеннойвыше формуле.

О*накопленная Итоговая=* 0.2*·*О*промежуточная 1+* 0.8*·*О*накопленная 2*

Способ округления накопленной оценки промежуточного (итогового) контроля в форме экзамена: арифметический.

[Сумма удельных весов должна быть равна единице: ∑mi = 1, при этом, **0,2 ≤ *m1 ≤* 0,8 (согласно Положению об организации контроля знаний, утвержденному УС НИУ ВШЭ от 24. 06.2011,протокол №26)**]

На пересдаче студенту не предоставляется возможность получить дополнительный балл для компенсации оценки за текущий контроль.

На зачете студент может получить дополнительный вопрос (дополнительную практическую задачу, решить к пересдаче домашнее задание), ответ, на который оценивается в 1 балл.

На экзамене студент может получить дополнительный вопрос (дополнительную практическую задачу, решить к пересдаче домашнее задание), ответ на который оценивается в 1 балл.

В диплом выставляет результирующая оценка по учебной дисциплине, которая формируется по следующей формуле:

*Орезульт =* 0.4*·Онакопл* + 0.6·*Оитоговый*

Способ округления результирующей оценки по учебной дисциплине: арифметический.

# Учебно-методическое и информационное обеспечение дисциплины

## Базовый учебник

1. Магда Ю.С. Ассемблер для процессоров Intel Pentium. –СПб.: Питер, 2006.-410с., ISBN 5-469-00662-X

## Основная литература

1. Внуков А.А. Лекции в электронной форме по курсу Информатика и программирование.– файл: Архитектура системы команд и язык Ассемблер.doc Лекции, 1-й модуль 2015. -145 с.
2. Пирогов В.Ю. ASSEMBLER. Учебный курс. - М.: Издатель Молгачева С.В., Издательство Нолидж, 2001. - 848 с., ил., ISBN 5-89251-101-4
3. Голубь Н.Г. Искусство программирования на Ассемблере. Лекции и упражнения. -2 изд., испр. и доп. –СПб.: ООО «ДиаСофтЮП». 2002.-656 с. ISBN 5-93772-056-3
4. Рудаков П.И., Финогенов К.Г. Язык ассемблера: уроки программирования. – ДИАЛОГ-МИФИ, 2001. -640 с. ISBN 5-86404-160-2
5. Ирвин К. Язык ассемблера для процессоров Intel. 3-е изд. / Пер. с анг. – М:. Издательский дом «Вильямс», 2005-ISBN: 5-8459-0779-9
6. Внуков А.А., Раев А.П., Можаитин Р.В. Организация ЭВМ, комплексов и систем. Учебное пособие — М.: Московский государственный институт электроники и математики, 2001. — 112 с. ISBN 5-230-16343-7
7. Внуков А.А. Лекции в электронной форме по курсу Архитектура ЭВМ и систем – файл: МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ИНСТИТУТ ЭЛЕКТРОНИКИ И МАТЕМАТИКИ -05.doc, лекции, 4 семестр 2013. -142 с.
8. Внуков А.А. Лекции в электронной форме по курсу Архитектура ЭВМ и систем – файл: Архитектура\_ЭВМ+система\_команд 2009.doc, лекции, 4 семестр 2009. -128 с.
9. Ivor Horton, Beginning Visual C++ 2010, Indianapolis : Wiley Publishing, Inc., 2011,
10. P.J. Deitel, H.M. Deitel, C++ How to Program, 7th Pearson Education, 2010,

## Дополнительная литература

1. IA-32 Intel Architecture Software Developer’s Manual, Intel® Corp. 2001
2. IA-32 Intel Architecture Optimization, Intel® Corp. 2001
3. Э. Кенинг, Б. Му, Эффективное программирование на С++ // М. : «Вильямс», 2002,
4. Айвор Хортон, Visual C++ 2010 полный курс, Москва : Вильямс, 2011
5. С. Мейерс, Эффективное использование С++: 50 рекомендаций по улучшению ваших программ и проектов // Москва : Питер, 2006,
6. С. Мейерс, Эффективное использование STL // Москва : Питер, 2002,
7. Г. Саттер, Решение сложных задач на С++ // Москва : Вильямс, 2002,
8. А. Александреску, Современное проектирование на С++ // Москва : Вильямс, 2002,

## Справочники, словари, энциклопедии

Не предусмотрены.

## Программные средства

Для успешного освоения дисциплины, студент использует следующие программные средства:

* Visual С++ 2010

## Дистанционная поддержка дисциплины

Не предусмотрены.

# Материально-техническое обеспечение дисциплины

Для проведения лекций необходима аудитория, оснащенная компьютером и проекционным оборудованием.

Для проведения практических занятий необходим компьютерный класс, оснащенный современным программным обеспечением, IDE для разработки программ: Visual С++ 2010.