**Topic Programming**

1. *What is programming?*

Computer programming encompasses a broad set of activities that include planning, coding, testing, and documenting. Most programmers participate in all of these phases of program development, but focus on the coding process. Software engineers tend to focus on designing and testing activities.

1. *What are the most common programming paradigms?*

The phrase programming paradigm refers to a way of conceptualizing and structuring the tasks a computer performs. Today’s most popular programming paradigms are:

|  |  |  |
| --- | --- | --- |
| Paradigm | Languages | Description |
| Event-driven | Visual Basic, C# | Focuses on selecting user interface elements and defining event-handling routines that are triggered by various mouse or keyboard activities |
| Procedural | BASIC, Ada, Pascal, Fortran, COBOL | Emphasizes linear steps that provide the computer with instructions on how to solve a problem or carry out a task |
| Object-oriented | Smalltalk, C++, Java, Scratch | Formulate programs as a series of objects and methods that interact to perform a specific task |
| Declarative | Prolog | Focuses on the use of facts and rules to describe a problem |

1. *What are the main development methodologies?*

A problem statement provides a minimal amount of planning, which is sufficient for only the simplest programs. A typical commercial application requires far more extensive planning, which includes detailed program outlines, job assignments, and schedules. Methodologies can be classified as predictive or agile.

A predictive methodology requires extensive planning and documentation up front. It allows little room for adaptation and change. Predictive methodologies are preferred for large software development projects. In contrast to predictive methodologies, an agile methodology focuses on flexible development and specifications that evolve as a project progresses. Programmers produce a subset of the entire project, show it to users, and then plan the next phase of development.

1. *What are programming languages classified?*

Programming languages can be divided into two major categories: low-level languages (machine languages and assembly languages) and high-level languages (based on human languages).

1. *OOP*

Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which can contain data and code. The data is in the form of fields (often known as attributes or properties), and the code is in the form of procedures (often known as methods).

OOP languages are diverse, but the most popular ones are class-based, meaning that objects are instances of classes, which also determine their types. A class is a template for a group of objects with similar characteristics.

In object-oriented jargon, inheritance refers to passing certain characteristics from one class to other classes. A superclass is any class from which attributes can be inherited. A subclass (or «derived class») is any class that inherits attributes from a superclass. The set of superclasses and subclasses that are related to each other is referred to as a class hierarchy.

A method is a segment of code that defines an action. A method is activated by a message, which is included as a line of program code, sometimes referred to as a «call». Polymorphism, sometimes called «overloading», is the ability to redefine a method in a subclass. It allows programmers to create a single, generic name for a procedure that behaves in unique ways for different classes. Polymorphism provides OO programs with easy extensibility and can help simplify program code.

1. *What is Artificial intelligence? How is AI used in our everyday life?*

Artificial intelligence (AI) is intelligence demonstrated by machines, as opposed to intelligence of humans and other animals. Example tasks in which this is done include speech recognition, computer vision, translation between (natural) languages, as well as other mappings of inputs.

AI applications include advanced web search engines (e.g., Google Search), recommendation systems (used by YouTube, Amazon, and Netflix), understanding human speech (such as Siri and Alexa), self-driving cars (e.g., Tesla), generative or creative tools (ChatGPT and AI art), automated decision-making, and competing at the highest level in strategic game systems (such as chess and Go).

1. *What are the main roles in game development?*

Producer, publisher, development team (designer, artist, programmer, level designer, sound engineer, tester)

1. *What technologies and languages are used for game development?*

Game programming is the creation of software that runs video games, including the game engine, user interface, physics engine, graphics, sound, and AI. The most commonly used programming languages in game programming include C++, Java, and Python, and game programmers must have a deep understanding of computer science and mathematics, as well as a strong sense of design and a passion for gaming.

1. *Robotic*

Robotic involves the development of technical systems based on various disciplines such as electronics, mechanics, cybernetics, and computer science. Robotics can be applied in different areas such as construction, industry, household, medical, aviation, military, space, and underwater. Building and programming robots are challenging tasks, but specialized programming languages such as ROS, Python, and MATLAB simplify the process. Robotics programming includes different approaches such as behavior-based robotics, control-based robotics, and evolutionary robotics.

1. *Robotic systems*

A robotic technological complex (abbreviated RTC) is a system consisting of one or more high-tech devices that operates autonomously and performs multiple cycles.

In general, there are three main **types** of robotic complexes:

1. RTC of the first type. They are formed on the basis of an industrial robot and auxiliary devices. In this case, the robot loads and unloads the equipment.
2. RTC of the second type. They consist of several industrial robots, as well as secondary devices. At the same time, industrial robots in this case, in addition to unloading and loading, transport parts between operations.
3. In the RTC of the third type, industrial robots, in addition to the above functions, are engaged in other work.

The **problems** of introducing a robotic complex of the first type to an enterprise are the same as when using “classic” industrial robots. In modern realities, because of them, RTC-1 often remain, as they say, out of business:

* complex programming;
* traditional industrial robots require a narrow knowledge of specialists;
* “sharpened” only for multi-part tasks;
* dangerous to humans;
* the company is forced to bear the costs of expensive in-house specialists or attract expensive ones from the outside;
* they are protected by a special robot cage, which is why they take up a lot of space in the workshop.

If you use collaborative robots in the structure of the RTC, then the company can get the following **advantages**:

* Saving money to pay for labor;
* High speed of production processes;
* Greater accuracy of work and a minimum of marriage;
* Energy efficiency;
* Lower processing costs;
* Improving the quality of management;
* Affordable prices for equipment;
* Mobility when performing different tasks.