Spring 2024-2025 Term 4

ИКНК/ФизМех

Ребдев Павел Александрович

Группа 5130904/30008

Active vocabulary: 30, Grammar structures: 4, Linkers: 12, Total: 675 words.

Monologue on *PROGRAM DESIGN AND COMPUTER LANGUAGES (UNIT 4)*

Step 1. Introduction 1. Start with a hook sentence that will attract the listener's attention, a quote, a proverb, etc. 2. Lead your speech steadily to the main part of your talk. 3. The introduction may consist of 3-6 sentences.	"The computer was born to solve problems that did not exist before." — Bill Gates. Indeed, a programming language is far more than just a tool—it is the bridge between human creativity and machine precision, a carefully crafted artificial language that enables us to perform computations, control external devices, and shape the digital world. From algorithms that drive modern AI to markup languages like HTML that structure the web, these languages are the backbone of technology. But what truly defines them? How have they evolved? And where are they headed? Let us explore the traits of programming languages, their design, their history, and their future.
Step 2. Important traits of a programming language 2.1. What are the traits important for constituting a programming language? 2.2. What are the contexts in which programming languages are used?	At their core, computer languages must possess expressive power—the ability to define and manipulate data structures, control the flow of execution, and classify languages by the computations they are capable of expressing. Unlike natural languages, which thrive on flexibility and nuance, programming languages demand a greater degree of precision and completeness. They rely on formal grammar to eliminate ambiguity, ensuring that a machine interprets instructions exactly as intended. Some languages, such as Turing complete languages, can express any set of algorithms, making them universal in capability. Others, like scripting languages, are specialized for particular domains, offering simplicity for hobbyists and efficiency for database developers. The diversity of contexts in which they operate is staggering—some are executed in a batch process without human interaction, while others facilitate interactive sessions where users dynamically input commands. Whether in software applications, speech recognition

	software, or local network environments, programming languages allow humans to communicate instructions to machines with unparalleled efficiency.
Step 3. Program design 3.1. How are programming languages different from natural languages? 3.2. Speak about the trends in the development of programming languages	While natural languages evolve organically, programming languages are designed from scratch with deliberate structure. They must expect one's intent to be understood without ambiguity, a stark contrast to the fluidity of human speech. Early languages were tied to the underlying hardware, requiring deep technical knowledge. However, modern languages solve problems using a higher level of abstraction, making them accessible to novices while remaining powerful for experts. One notable trend is the rise of domain-specific programming languages, tailored for niche applications. Some older languages fall into disuse, while others are altered to meet new needs. For instance, scripting languages have surged in popularity due to their flexibility, whereas assembly languages remain indispensable for low-level hardware control. Additionally, languages increasingly are combined with other languages, blending paradigms to enhance functionality.
Step 4. History of programming languages 4.1. The predecessors of programming languages. What were they? 4.2. Speak about the generations of programming languages.	The predecessors of programming languages were not digital but mechanical—looms controlled by punch cards, player piano scrolls encoding musical instructions, and early punch cards used in computation. These innovations laid the groundwork for the theory of computation, formalized through lambda calculus and Turing machines, which became the foundation of modern programming. The first generation of languages consisted of machine code, directly executed by hardware. This was followed by assembly languages, which introduced macro instructions for improved readability. The development of high-level programming languages, such as those used in UNIVAC, marked a revolution, enabling programmers to write code that was less tied to the underlying hardware. Over time, languages continued to undergo modification, with some superseding older ones, while others evolved into descendants with enhanced features.

Step 5. CREATIVE THINKING Introduce your own extra idea(s) on program design and computer languages that hasn't/haven't been mentioned before. Substantiate your choice.	One underexplored concept is the idea of self-adapting languages—systems that are chained together with AI to refine their own syntax based on real-world usage. Such languages could autonomously optimize for efficiency, reducing the need for manual updates. Another possibility is context-aware programming, where languages dynamically adjust their behavior depending on the program's inputs or environment. These innovations could redefine program design, making coding more intuitive and adaptive.
Step 6. Conclusion Summarise the ideas of steps 2,3,4,5.	From their fundamental traits of programming languages to their rich and complex history, these constructs have revolutionized human-machine interaction. They classify languages by the computations they are capable of expressing, ensuring precision across countless applications. As technology advances, programming languages will continue to evolve—altered to meet new needs, yet always grounded in the theory of computation. Whether you are a hobbyist, a novice, or a seasoned developer, understanding these languages empowers you to shape the future. After all, in the realm of code, the only true limit is imagination.