Tutorial Link https://codequotient.com/tutorials/Introduction to Programming/5a2ccf816dc3de029e488809

TUTORIAL

Introduction to Programming

Chapter

1. Introduction to Programming

Today, we use computers in almost every part of our life. Think about some of the different ways that people use computers. In school, students use computers for tasks such as writing papers, searching for articles, sending email, and participating in online classes etc. At work, people use computers to analyze data, make presentations, conduct business transactions etc. At home, people use computers for tasks such as paying bills, shopping online, communicating with friends and family, playing computer games etc. Also our cell phone, microwave ovens, refrigerators, washing machines and other equipment's are all using computers in direct or indirect forms. The uses of computers are almost limitless in our everyday lives.

Computers can do such a wide variety of things because they can be programmed. This means that computers are not designed to do just one job, but to do any job that their programs tell them to do. A program is a set of instructions that a computer follows to perform a task. We all are familiar with word processing software's like Microsoft word, open office documents etc. These are word processing programs those allows you to create, edit, and print documents with your computer. Programs are commonly referred to as software. Software is essential to a computer because it controls everything the computer does. All of the software that we use to make our computers useful is created by individuals working as programmers or software developers. A programmer, or software developer, is a person with the training and skills

necessary to design, create, and test computer programs. Computer programming is an exciting and rewarding career.

Programs that make a computer useful for everyday tasks are known as application software. These are the programs that people normally spend most of their time running on their computers. A computer will store the data and instructions in its memory. Computer memory is organized as the collection of bits. Bit is the smallest part of memory which can store two situations called on or off. Memory is generally managed by bytes (Where a byte is a combination of 8 bits). A computer's CPU can only understand instructions that are written in machine language. Because people find it very difficult to write entire programs in machine language, other programming languages have been invented.

The CPU is an electronic device that is designed to do specific things. In particular, the CPU is designed to perform operations such as the following:

• Reading a piece of data from main memory

• Adding two numbers

• Subtracting one number from another number

• Multiplying two numbers

• Dividing one number by another number

• Moving a piece of data from one memory location to another

• Determining whether one value is equal to another value

As you can see from this list, the CPU performs simple operations on pieces of data. The CPU does nothing on its own, however. It has to be told what to do, and that's the purpose of a program. A program is nothing more than a list of instructions that cause the CPU to perform operations. Each instruction in a program is a command that tells the CPU to perform a specific operation. To a CPU, however, an instruction to perform an operation is written in

0s and 1s because CPUs only understand instructions that are written in machine language and machine language instructions always have an underlying binary structure.

Although a computer's CPU only understands machine language, it is impractical for people to write programs in machine language. For this reason, assembly language was created in the early days of computing as an alternative to machine language. Instead of using binary numbers for instructions, assembly language uses short words that are known as mnemonics. For example, in assembly language, the mnemonic add typically means to add numbers, mul typically means to multiply numbers, and mov typically means to move a value to a location in memory. When a programmer uses assembly language to write a program, he or she can write short mnemonics instead of binary numbers.

Although assembly language makes it unnecessary to write binary machine language instructions, it is not without difficulties. Assembly language is primarily a direct substitute for machine language, and like machine language, it requires that you know a lot about the CPU. Assembly language also requires that you write a large number of instructions for even the simplest program. Because assembly language is so close in nature to machine language, it is referred to as a low-level language.

So, a new generation of programming languages known as high-level languages began to appear. A high-level language allows you to create powerful and complex programs without knowing how the CPU works, and without writing large numbers of low-level instructions. In addition, most high-level languages use words that are easy to understand.

Each high-level language has its own set of predefined words that the programmer must use to write a program. The words that make up a high-level programming language are known as keywords or reserved words. Each keyword has a specific meaning, and cannot be used for any other purpose.

Because the CPU understands only machine language instructions,

programs that are written in a high-level language must be translated into machine language. Depending on the language that a program has been written in, the programmer will use either a compiler or an interpreter to make the translation. A compiler is a program that translates a high-level language program into a separate machine language program. The machine language program can then be executed any time it is needed.