High-Level Languages

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.

In the 1950s, 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. For example, if a programmer were using COBOL (which was one of the early high languages), he or she would write the following instruction to display the message Hello world on the computer screen:

DISPLAY "Hello world"

Python is a modern, high-level programming language. In Python you would display the message Hello world with the following instruction:

print 'Hello world'

Doing the same thing in assembly language would require several instructions, and an intimate knowledge of how the CPU interacts with the computer’s output device. high-level languages allow programmers to concentrate on the tasks they want to perform with their programs rather than the details of how the CPU will execute those programs. Since the 1950s, thousands of high-level languages have been created. Table below lists several of the more well-known languages.

Advantages:

1) Easy to write and understand 2) Easy to isolate an error 3) Machine independent language 4) Easy to maintain 5) Better readability 6) Low Development cost 7) Easier to document 8) Portable

Disadvantages:

1) Needs translator 2) Requires high execution time 3) Poor control on hardware 4) Less efficient

Language	Description
Ada	Ada was created in the 1970s, primarily for applications used by the U.S. Department of Defense. The language is named in honor of Countess Ada Lovelace, an influential and historic figure in the field of computing.
BASIC	**B**eginners **A**ll-purpose **S**ymbolic **I**nstruction **C**ode is a general-purpose language that was originally designed in the early 1960s to be simple enough for beginners to learn. Today, there are many different versions of BASIC.
FORTRAN	**FOR**mula **TRAN**slator was the first high-level programming language. It was designed in the 1950s for performing complex mathematical calculations.
COBOL	**C**ommon **B**usiness-**O**riented **L**anguage was created in the 1950s, and was designed for business applications.
Pascal	Pascal was created in 1970, and was originally designed for teaching programming. The language was named in honor of the mathematician, physicist, and philosopher Blaise Pascal.
C and C++	C and C++ (pronounced “c plus plus”) are powerful, general-purpose languages developed at Bell Laboratories. The C language was created in 1972 and the C++ language was created in 1983.
C#	Pronounced “c sharp.” This language was created by Microsoft around the year 2000 for developing applications based on the Microsoft .NET platform.
Java	Java was created by Sun Microsystems in the early 1990s. It can be used to develop programs that run on a single computer or over the Internet from a web server.
JavaScript	JavaScript, created in the 1990s, can be used in web pages. Despite its name, JavaScript is not related to Java.
Python	Python, is a general-purpose language created in the early 1990s. It has become popular in business and academic applications.
Ruby	Ruby is a general-purpose language that was created in the 1990s. It is increasingly becoming a popular language for programs that run on web servers.
Visual Basic	Visual Basic (commonly known as VB) is a Microsoft programming language and software development environment that allows programmers to create Windows based applications quickly. VB was originally created in the early 1990s.

Data Types & Arithmetic Expressions

Objective is to be able to list, describe, and use the C basic data types (C as example of educational language), to be able to create and use variables and constants and to be able to use simple input and output statements and Learn about type conversion.

Data Types

A type defines by the following: A set of **values** and A set of **operations** "C" offers three basic data types:
Integers defined with the keyword **int**
Characters defined with the keyword **char**
Real or floating-point numbers defined with the keywords **float** or **double**.
Its importance lies in improve code readability and A common convention, or good practice, is to name constant variables using upper case letters, to make constant variables clearly visible in code.
There are two simple ways in C to define constants:

- Using **#define** preprocessor.
- Using **const** keyword.

1. Integers

positive or negative whole number**s** Thre are three types of integers:
“int”,
“short int” (which can be abbreviated “short”)
“long int” (which can be abbreviated “long”).
Example

const int MAXRATE = 10; /*int constant*/

2. Real numbers

There is an infinite number of real numbers, but they are represented with a finite number of bits in the computer. There are two main types: “float” and “double”. Real numbers defined with “double” are represented with a size double of those declared as “float”. The difference is the amount of precision used to represent the numbers internally.
Example float:

const float PI = 3.14; /*Real constant float */

Example double:

const double SPEED_OF_SOUND= 761.207; // Miles/hour (sea level) const double SECONDS_PER_HOUR = 3600.0; // Secs/hour

3. Characters and strings

Character:The variables of type character are declared as “char”. To refer to a character, the symbol must be surrounded by **simple quotes**: 'M'. Characters are internally represented as numbers and the C language allows arithmetic operations with them such as 'M' + 25.
Example

const char MYCHARACTER = 'A'; /*char constant*/

String:The strings are represented as tables of “char”. The library functions to manipulate strings all assume that the last byte of the chain has value zero. The strings are written in the program surrounded by **double quotes** and contain the value zero at the end.
Example

const char MYAREA[10] = "Tysons Corner"; /*string constant*/

Arithmetic Expressions

- As in most languages, C programs specify computation in the form of arithmetic expressions that closely resemble expressions in mathematics.
- The most common operators in C are the ones that specify arithmetic computation:

Operator	Name	Example	Equivalent construct
+=	Addition assignment	x += 4;	x = x + 4;
-=	Subtraction assignment	x -= 4;	x = x - 4;
*=	Multiplication assignment	x *= 4;	x = x * 4;
/=	Division assignment	x /= 4;	x = x / 4;
%=	Remainder assignment	x %= 4;	x = x % 4;

Binary Operators:

- Operators in C usually appear between two subexpressions, which are called its **operands**. **Operators** that take two operands are called binary operators: Operand operator Operand A + B

Unary Operator:
The - operator can also appear as a **unary operator**, as in the expression -x, which denotes the negative of x.
Precedence rules for arithmetic operators:

- ### Shorthand operators

A shorthand operator is a shorter way to express an expression. Shorthand operators +=, -=, *=, /= and *=

- Frequent expressions: x is a variable in the program
- Special Statements: Increment and decrement operators:

Operator	Name	Example	Equivalent construct
++	Increment	x++;	x = x + 1;
--	Decrement	x--;	x = x - 1;

Type conversions

A **type conversion** (also known as type **casting**, and type **coercion**) is a conversion of one data type to another, such as an int to a double. It is needed when the types of an expression are not compatible: There are two types of type conversions:

- **Implicit conversion (Also known as coercion):** When the compiler automatically performs several common conversions between int and double types.
- **Explicit conversion (Also known as casting):** When the user decides the type of the conversion (the desired type)

For assignment =, the right-side type is converted to the left side type. oint-to-double conversion is straightforward

25 becomes 25.0.

odouble-to-int conversion just drops the fraction:

4.9 becomes 4.

Arithmetic Expressions: Conversions are implicitly performed to cast their values to a common type, if the user does not specify any casting.

Numbering Systems:

The computer work with numbers only, letters and symbols also the numbers transfer to specific numbers input to computer and process then output the results which transfer from machine language to language understand by humans, display on different output devices. The most numbering systems used are:

1. **Decimal System**
2. **Binary System**
3. **Octal System**
4. **Hexadecimal System**

Following the properties of Numbering Systems:

1. Decimal System

The properties of decimal system as follows:

a) Base is 10
b) set of used symbols are:(0,1,2,3,4,5,6,7,8,9)

Example: The number in decimal system ($29029.98)_{10}$ is compute as follows:

$2 * 10^4 + 9 * 10^3 + 0 * 10^2 + 2 * 10^1 + 9 * 10^0 + 9 * 10^{-1} + 8 * 10^{-2}=29029.98$

Number	Number/2	Submit
25	12	1
12	6	0
6	3	0
3	1	1
1	0	1

2. Binary system

The properties of Binary system as follows:

a) Base is 2
b) Set of used symbols are:(0,1)

Example: convert the number $(10101.01)_2$ From binary system to decimal system

$1 * 2^4 + 0 * 2^3 + 1 * 2^2 + 0 * 2^1 + 1 * 2^0 + 0 * 2^{-1} + 1 * 2^{-2}= (21.25)_{10}$

Example: convert the number $(25)_{10}$ from decimal system to binary system

The number in decimal system $(25)_{10}$ =$(11001)_2$ in binary system.

Number	Number/8	Submit
30	3	6
3	0	3

3. Octal System

The properties of octal system as follows:

a) Base is 8
b) Set of used symbols are:(0,1,2,3,4,5,6,7)

Example: convert the number:$(74211.03)_8$ from octal system to decimal system:

$7 * 8^4 +4 * 8^3 + 2 * 8^2 + 1 * 8^1 + 1 * 8^0 + 0 * 8^{-1} + 3 * 8^{-2}=(30857.05)_{10}$

Example: convert the number $(30)_{10}$ from decimal system to octal system

The number in decimal system $(30)_{10}$ =$(36)_8$ in octal system.

Number	Number/16	Submit
50	3	2
3	0	3

4. Hexadecimal System

The properties of hexadecimal system as follows:

a) Base is 16
b) Set of used symbol are:(0,1,2,3,4,5,6,7,8,9 ,A,B,C,D,E,F) Where A=10, B=11, C=12, D=13, E=14, F=15

Example: convert the number from hexadecimal to decimal system $(F3B05.A2)_{16}$

$15 * 16^4 +3 *16^3 + 11 * 16^2 + 0 * 16^1 + 5 * 16^0 +10 * 16^{-1} + 2 * 16^{-2}=(998149.6)_{10}$

Example: convert the number $(50)_{10}$ from decimal system to hexadecimal system

The number in decimal system $(50)_{10}$ =$(32)_{16}$ in hexadecimal system.