Computer Languages

Professor Dr. Md. Ismail Jabiullah
Department of CSE
Daffodil International University

Learning Objectives

- **In this Lecture you will learn about:**
- Computer languages or programming languages
- Three broad categories of programming languages – machine, assembly, and high-level languages
- Commonly used programming language tools such as assembler, compiler, linker, and interpreter
- Concepts of object-oriented programming languages
- Some popular programming languages such as FORTRAN, COBOL, BASIC, Pascal, C, C++, C#, Java, RPG, LISP and SNOBOL
- Related concepts such as Subprogram, Characteristics of a good programming language, and factors to consider while selecting a language for coding an application

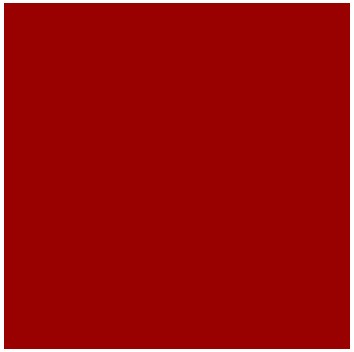
Broad Classification of Computer Languages

- Machine language
- Assembly language
- High-level language

Machine Language

- Only language of a computer understood by it without using a translation program
- Normally written as strings of binary 1s and 0s
- Written using decimal digits if the circuitry of the computer being used permits this

Advantages & Limitations of Machine Language



Advantages

- Can be executed very fast

Limitations

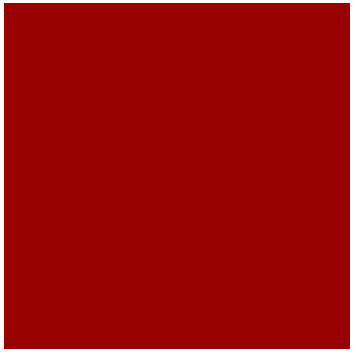
- Machine Dependent
- Difficult to program
- Error prone
- Difficult to modify

Assembly/Symbolic Language

- Programming language that overcomes the limitations of machine language programming by:

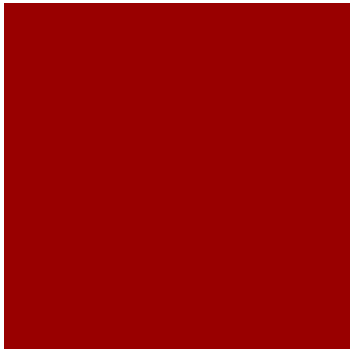
- Using alphanumeric mnemonic codes instead of numeric codes for the instructions in the instruction set e.g. using ADD instead of 1110 (binary) or 14 (decimal) for instruction to add
- Allowing storage locations to be represented in form of alphanumeric addresses instead of numeric addresses e.g. representing memory locations 1000, 1001, and 1002 as FRST, SCND, and ANSR respectively
- Providing pseudo-instructions that are used for instructing the system how we want the program to be assembled inside the computer's memory e.g. START PROGRAM AT 0000; SET ASIDE AN ADRESS FOR FRST

Advantages of Assembly Language Over Machine Language



- Easier to understand and use
- Easier to locate and correct errors
- Easier to modify
- No worry about addresses
- Easily relocatable
- Efficiency of machine language

Limitations of Assembly Language



- Machine dependent
- Knowledge of hardware required
- Machine level coding

High-Level Languages

- Machine independent
- Do not require programmers to know anything about the internal structure of computer on which high-level language programs will be executed
- Deal with high-level coding, enabling the programmers to write instructions using English words and familiar mathematical symbols and expressions

Advantages of High-Level Languages

- Machine independent
- Easier to learn and use
- Fewer errors during program development
- Lower program preparation cost
- Better documentation
- Easier to maintain

Limitations of High-Level Languages

- Lower execution efficiency
- Less flexibility to control the computer's CPU, memory and registers

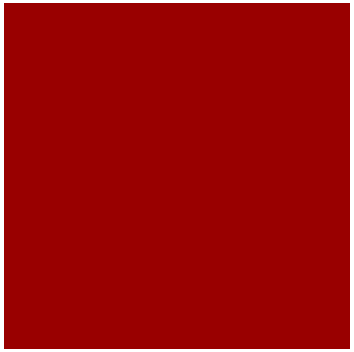
C

- Developed in 1972 at AT&T's Bell laboratories, USA by Dennis Ritchie and Brian Kernighan
- Standardized by ANSI and ISO as C89, C90, C99
- High-level programming languages (mainly machine independence) with the efficiency of an assembly language
- Language of choice of programmers for portable systems software and commercial software packages like OS, compiler, spreadsheet, word processor, and database management systems

C is Middle Level Language

- There are following reason that C is called Middle Level Language as:
  - C programming language behaves as high level language through function, it gives a modular programming and breakup, increased the efficiency for resolvability.
  - C programming language support the low level language i.e. Assembly Language.
  - C language also gives the facility to access memory through pointer.
  - Its combines the elements of high-level languages with the functionalism of assembly language.
- So, C language neither a High Level nor a Low level language but a **Middle Level Language**.

Variables in C



Topics

- What is Variable
- Naming Variables
- Declaring Variables
- Using Variables
- The Assignment Statement

What Are Variables in C?

- Variables are the names that refer to sections of memory into which **data can be stored**.
- **Variables** in C have the same meaning as variables in algebra. That is, they represent some unknown, or variable, value.

$$x = a + b$$

$$z + 2 = 3(y - 5)$$

- Remember that variables in algebra are represented by a single alphabetic character.

Naming Variables

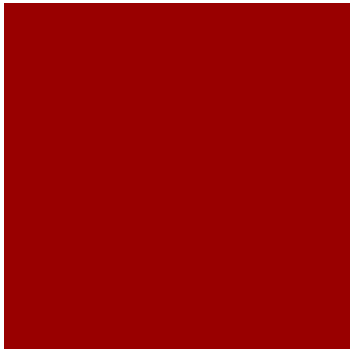
- Rules for variable naming:
  - Can be composed of letters (both uppercase and lowercase letters), digits and underscore only.
  - The first character should be either a letter or an underscore(not any digit).
  - Punctuation and special characters are not allowed except underscore.
  - Variable name should not be keywords.
  - names are case sensitive.
  - There is no rule for the length of a variable name. However, the first 31 characters are discriminated by the compiler. So, the first 31 letters of two name in a program should be different.

Naming Conventions

- C programmers generally agree on the following **conventions** for naming variables.
  - Begin variable names with lowercase letters
  - Use meaningful identifiers
  - Separate "words" within identifiers with underscores or mixed upper and lower case.
  - Examples: surfaceArea surface_Area surface_area
  - Be consistent!
- Use all uppercase for **symbolic constants** (used in **#define** preprocessor directives).Examples:

```
#define PI 3.14159
#define AGE  52
```

Reserved Words (Keywords) in C



auto	break	int	long
register	return	short	signed
size of	static	struct	switch
Typedef	union	unsigned	void
Volatile	while	case	char
const	continue	default	do
double	else	enum	extern
float	for	goto	if

Declaring Variables

- Before using a variable, you must give the compiler some information about the variable; i.e., you must **declare** it.
- C has three basic predefined data types:
- Integers (whole numbers): **int**

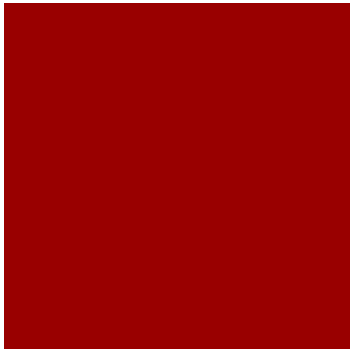
  int length = 7 ;
- Floating point (real numbers): **float, double**

  float diameter = 5.9 ;
- Characters: **char**

  char initial = ‘A’ ;

A Simple C Program

```
#include<stdio.h>
int main()
 {
printf("Hello World");
}
```



Sum of two numbers

```
#include <stdio.h>
int main( )
{
int num1, num2, sum;
 printf("Enter two integers: ");
scanf("%d %d",&num1,&num2); /* Stores the two integer
   entered by user in variable num1 and num2 */
sum=num1+num2; /* Performs addition and stores it in
   variable sum */
printf("Sum: %d",sum); /* Displays sum */
return 0;
}
```

Printf() and Scanf() functions

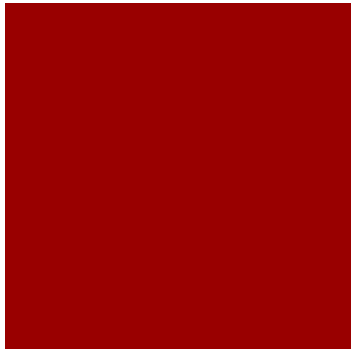
- printf() and scanf() functions are inbuilt library functions in C which are available in C library by default.
- These functions are declared and related macros are defined in “stdio.h” which is a header file.
- We have to include “stdio.h” file as shown in below C program to make use of these printf() and scanf() library functions.

printf() function

- **C printf() function:**
  - The **printf statement allows you to send output to standard out.** For us, standard out is generally the screen.
  - printf() function is used to print the "character, string, float, integer, octal and hexadecimal values" onto the output screen.
  - We use printf() function with %d format specifier to display the value of an integer variable.
  - Similarly %c is used to display character, %f for float variable, %s for string variable, %lf for double and %x for hexadecimal variable.
  - To generate a newline,we use "\n" in C printf() statement.
- **Note:**
  - C language is case sensitive. For example, printf() and scanf() are different from **P**rintf() and **S**canf(). All characters in printf() and scanf() functions must be in lower case.

scanf() function

- **scanf() function:**
  - The **scanf function allows you to accept input from standard in**, which for us is generally the keyboard.
  - scanf() function is used to read character, string, numeric data from keyboard
  - Consider below example program where user enters a character. This value is assigned to the variable "ch" and then displayed.
  - Then, user enters a string and this value is assigned to the variable "str" and then displayed.



Thank You