APS105 – Computer Fundamental

Computer system

- Can be divided into 2 entities:

***Hardware**:*

- Electronic Components and Electromechanical devices that comprise the physical entity of a Computer.
    - Central processing unit (CPU): ALU, Registers
    - Memory: RAM (Random Access Memory), ROM (Read Only Memory)
    - I/O devices: Keyboard, Mouse, Display, printer, scanner.

***Software**:*

- Instructions and data that computer manipulates to perform various data-processing tasks.

Machine Language is what hardware understand, manipulate, and can respond to.

High Level Language (HLL) contains easy-to-read syntax that is closer to human language and can be converted (compiled) into a machine language. Most of common programming languages are considered as high-level languages.

How does a computer work? Similar to turning on/off a light.
Consider:

Bit: 1/0
Byte: 8 bits

Task: is anything that executes/performs something, e.g. calculate average, calculate income tax, control airplane operation.

- F35 (Joint Strike Fighter)
  - Approximately has 1.1 million lines of code
  - Has 3 shoe box computers
  - Replaces 6 different airplanes
- BMW M3/M5 Series
  - Can perform breaking by using Engine Power.

Execution of a Task:

1- Write a program and save it as *name**.c*** in a file, e.g. *lab1.c*
2- Compile the program to generate a machine language, i.e. an executable file (name.exe)
3- Run the executable.

Compiler:

- Is a computer program that translates text written in a computer language (e.g. C language) into another computer language (e.g. machine language, 0's and 1's).

Integrated Development Environment (IDE):

- Need a number of software applications to write/execute a program such as editor, compiler, debugger, linker, library functions, etc.
- IDE provides all programming tools needed in one software, e.g. Codelite.

C Program:

- Is a collection of functions, keywords, variables, operators, expressions, statements, and different data types performing one or more tasks.

Example:

```
/*    Definition: This is my first program.
      Author:     HST
      Date:       today's date
*/
#include   <stdio.h>

main()
{
      printf("Hello!");
}
```

Example:

```
/*************************************************************
**    Definition:      This is my second C program. Prints two
**                     messages to the display.
**    Author:          HST
**    Date:            2030 alien calendar
*************************************************************/

#include <stdio.h>
main ( ){
      printf("Hello!\n");
      printf("Welcome to this course!\n");
}
```

- After execution of the above program the following lines appear on the screen.

1st line: Hello!
2nd line: Welcome to this course!

Comments:

- Are to document the program.
- Provide information for the programmer and compiler ignores them.
- Use "/*" at the start and "*/" at the end.

Functions:

- Are written by programmers or stored in the C library.
- Indicate the name of library when the library functions are used.
- Compiler includes details of the function in the program.

#include <stdio.h>

- Includes a definition or a function that can be found in the standard input/output library, e.g. "printf" function.

main ( ){ }

- C compiler needs to know where to start executing the program.
- "main" is the first function to be executed.
- All 'C' programs MUST start with "main" and only include one "main" function.
- Parentheses must appear after the "main**()**".
- Opening brace indicates where the "main" code starts.
- Closing brace indicates where the "main" code ends.

Note: Always use parentheses and braces in pairs.

C Statement: Controls the flow of the execution of a program.

- *Single Statement:* One single statement ending with a semicolon, e.g. x = 1 + 2;
- *Compound Statement:* A group of statements (more than one) that are enclosed by braces, e.g.

```
{
        x = 1;
        y = 2;
        z = x + y;
}
```

C Operators

- Are used to perform specific operations on the variable(s)

Arithmetic Operators

Indicate arithmetic operations such as addition, subtraction, multiplication, division, etc.

TABLE 1: ARITHMETIC OPERATORS IN C

OPERATOR	DENOTE	EXAMPLE	WHAT IT DOES
+	Addition	2+8	Add numbers & return the result 2+8 returns 10
-	Subtraction	12-8	Subtract numbers & return the result 12-8 returns 4
*	Multiplication	5*8	Multiply numbers & return the result 5*8 returns 40
/	Division	40/5	Divide numbers & return the result 40/5 returns 8
%	Remainder (modulo)	7%3	Divide numbers & return the remainder 7%3 Returns 1

Data types in C

Major data types are:

- Numbers: any numerical value, e.g. “3”
- Characters: any item from set of characters, e.g. “R”
- Strings: a combination of characters, e.g. “University”
- Void: any expression that does not have any value

Numbers in C

Two general categories:

- Integers
  - Unsigned – Only positive integers
  - Signed – Positive and negative integers

  Note: If sure that never negative numbers occur then use Unsigned, e.g. Number of students in a class.
- Floats

TABLE 2: NUMBERS IN C

DATA TYPE	PURPOSE	BYTES	RANGE
int	integer	*4*	*-2,147,483,648 to +2,147,483,647*
unsigned int	Unsigned integer	*4*	*0 to 4,294,967,295*
float	floating point	*4*	*3.4E+/-38*
double	double float	*8*	*1.7E+/-308*
long double	long double float	*16*	*1.7E+/-4932*

Use the data type that conserves memory.

Note: Suppose only one byte is required to declare a number. Considering that doubles are 8 bytes and long doubles are 16 bytes. If long doubles are used then 8 bytes are wasted => Not an efficient program.

Declaring Variables in C

All variables in 'C' must be declared so that the complier knows the:

1. Variable name, and
2. Type

So, enough memory can be allocated before the variable is used.

Data types are:

- char
- int
- float, double, long double
- void

How to declare?

```
/*Declaring variables of type character.*/      ← This Denotes a comment in C
char            aCharacter;
char            letter;

/*Declaring variables of type integer.*/
int             anInteger;
int             number;

/* Declaring variables of type float.*/
float           floatingPointNumber;
float           average;

/*The following is also valid:*/
int             age, number, mark;

/*************************************************************************
* Example: This program performs an addition. The sum is printed to the display.
*************************************************************************/
#include <stdio.h>
main ( ){
        int     x, y, sum;
        x = 1;
        y = 2;
        sum =  x + y;
        printf("Hello Again!\n");
        printf("Here is the sum = %d\n", sum);

}
```

1st line: Hello Again!

2nd line: Here is the sum = 3

printf() Function

- Prints information into the standard output (display).
- Arguments may include:
  - Message: Strings are specified using double quote "This is a string".
  - Variables' values.
  - Format Specifiers: Provides information for the printf() function as how to print its arguments and starts with "%".

TABLE 3: FORMAT SPECIFIERS FOR PRINTF( ) FUNCTIONS

SPECIFIER	PURPOSE
%d	decimal integer
%f	Signed floating point
%lf	Signed double
%e	Signed floating point using e notation
%c	A single character
%s	Strings

Escape Sequences

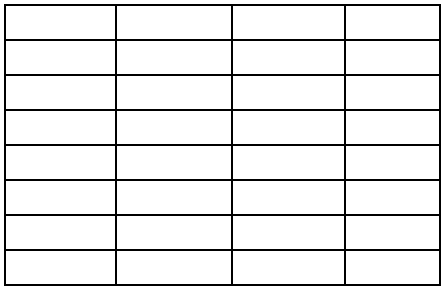
Are used to format the print out. "\" symbol is referred to as the escape character and is used to signify an escape sequence.

TABLE 5: ESCAPE SEQUENCES

SEQUENCE	PURPOSE
\n	New line
\t	Tab
\"	To print a double quote
\\	To print a backslash
\a	Audible alarm

Main Memory:

- Is the RAM (Random Access Memory).
- Is byte addressable.
- Is typically represented as:



- Suppose declaring variables:
  int x;
  x = 3;

- To get the address of x use "&", e.g.
  &x gives the address 1000

scanf() Function

- Resides in <stdio.h>
- Gets input from the standard input.
- Uses variable address.
- e.g.

```
scanf("%lf", &var);              /* get a value for var from the user */

%lf              tells the program the type of the data, e.g. double
&var             the address of var in memory.

scanf("%lf %d %c", &var1, &var2, &var3);
                                 /* get values for var1, var2, var3 */
                                 /* var1 is double */
                                 /* var2 is integer */
                                 /* var3 is character */
```

NOTE: **scanf( ) does not use the variable name. It uses the variable address.**

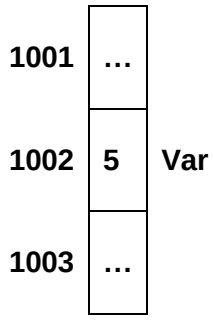
Example:

```
/* get a value from the user */
#include <stdio.h>
main( ){
        int     var;

        scanf("%d", &var);
        printf("Here is your entery: %d\n", var);
}

%d              tells the program the type of the data, i.e. an integer
&var            the address of var in memory.
```

Note: Use address operator (&) to get the address of variable.
For example: Var=5;



- **Var** represents the value of variable (5); the content of the memory.
- **&Var** represents the address of the variable (1002).

Example:

```
/*********************************************
 TO CALCULATE GPA
*********************************************/

#include <stdio.h>

main( ){
        int             numOfCourses=3;
        double          APS105, ECE110, APS112;
        double          average;

/* Input the first course grade   */
        printf("Please enter your grade for APS105: ");
        scanf("%lf", &APS105);

/* Input the second course grade  */
        printf("Please enter your grade for ECE110: ");
        scanf("%lf", &ECE110);

/* Input the third course grade  */
        printf("Please enter your grade for APS112: ");
        scanf("%lf", &CMTH140);

/* Calculate and display the GPA */
        average=( APS105 + ECE110 + APS112) / numOfCourses;

        printf("Your GPA is: %lf \n", average);
}
```