C PROGRAMMING LANGUAGE

Mansoura

Open Source

Course Outline

Course Assessment

- 40% : Lab
- 60%: Exam

Introduction

- Programing
- Hardware
- Software
- Software Life Cycle
- C History
- C Life Cycle.
- First C Program
- Variables

Programing

- Programing is the process to create a program
- Program is set of ordered instructions that enable a computer to carry out a specific task to solve humans problems

Hardware

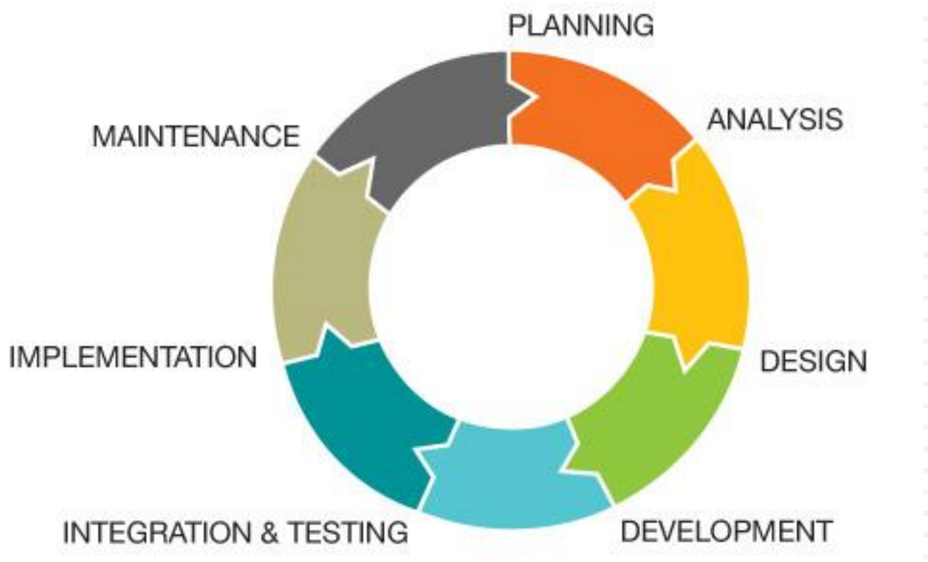
- Any physical components is hardware
- Computer hardware can only understand electrical signals (ON & Off)

Software

- Software is a collection of computer programs and related data that provide the instructions for telling a computer what to do and how to do

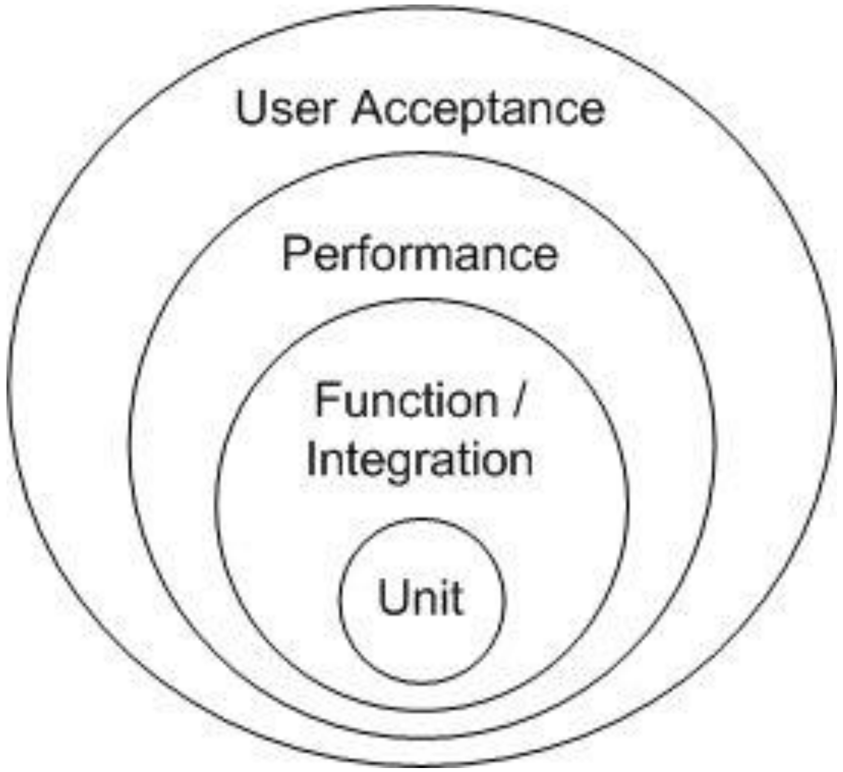
Software Life Cycle

- Gathering Requirement
- Analysis
- Design
- Development
- Testing
- Deployment
- Maintenance



Testing

- Unit Test
- Functional Test
- Performance Test
- Stress Test
- User Acceptance Test



Deployment

- Direct
- Parallel
- Phase

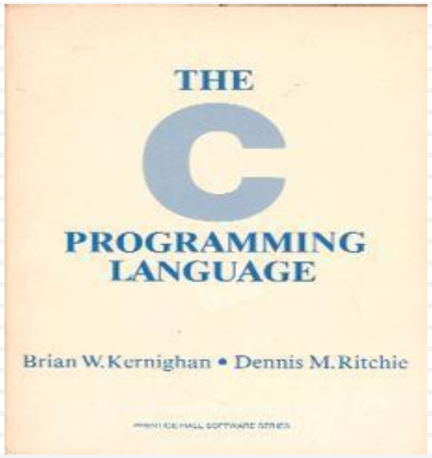
C History

1972: C Developed by Dennis Ritchie at Bell Labs.

1978: The C Programming Language by Brian Kernighan and Dennis Ritchie K & R C standard

1989: ANSI C standard

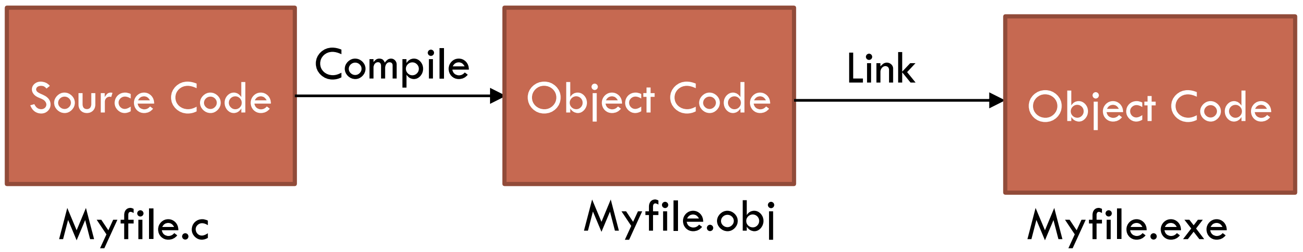
1999: C99 standard



Why Learn C

- C is a widely used portable high level programming language
- High level, but also let's your program closer to the metal.
- Used for embedded programming
- Many language have a c pedigree
- Useful for all applications.
- C is the native language for unix.

C Program Life Cycle



IDE

Is a software that is used to edit , compile and link a programs to produce executable programs as example TC , netbeans , vstdio.

We will use **Turbo C++ 4.0 Windows 7 Windows 8 64Bit Version**

Compiler & Interpreter

- Compilers and interpreters are used to convert the code of high level language into machine language. The high level program is known as source program and the corresponding machine level program is known as object program. Although both compilers and interpreters perform the same task but there is a difference in their working

Compiler & Interpreter

Compiler

A compiler searches all the errors of a program and lists them. If the program is error free then it converts the code of program into machine code and then the program can be executed by separate commands.

Interpreter

An interpreter checks the errors of a program statement by statement. After checking one statement, it converts that statement into machine code and then executes that statement. The process continues until the last statement of program occurs.

Compiler & Interpreter

Compiler

- Compiler takes **entire** program as input
- Intermediate object code is **generated**
- Program need not be **compiled** every time
- **Errors** are displayed after **entire program** is checked
- Example : C , C++

Interpreter

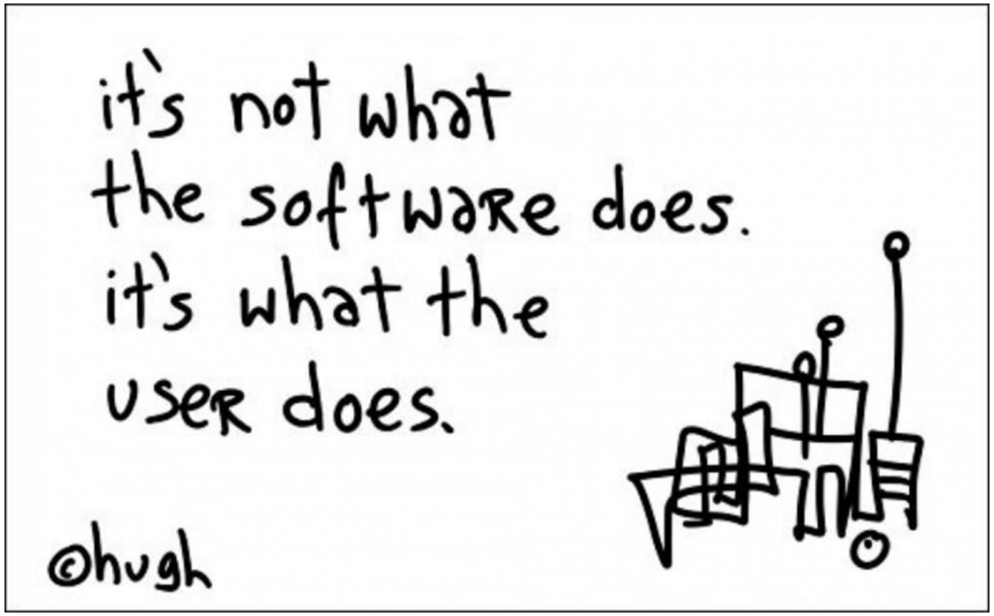
- Interpreter takes **single** instruction as input.
- **No** Intermediate object code is **generated.**
- every time higher level program is converted into lower level program.
- **Errors** are displayed for **every instruction** interpreted (if any)
- Example: Basic

Static & Dynamic Linking

Static	Dynamic
Extension .lib	Extension .dll
The data copied on every executable file	Only reference to the library on the executable file
Big in size executable file	Small in size Executable file
The Executable file run without need of the library file	Executable file need library to run

Users

- Developer
- Tester
- End User



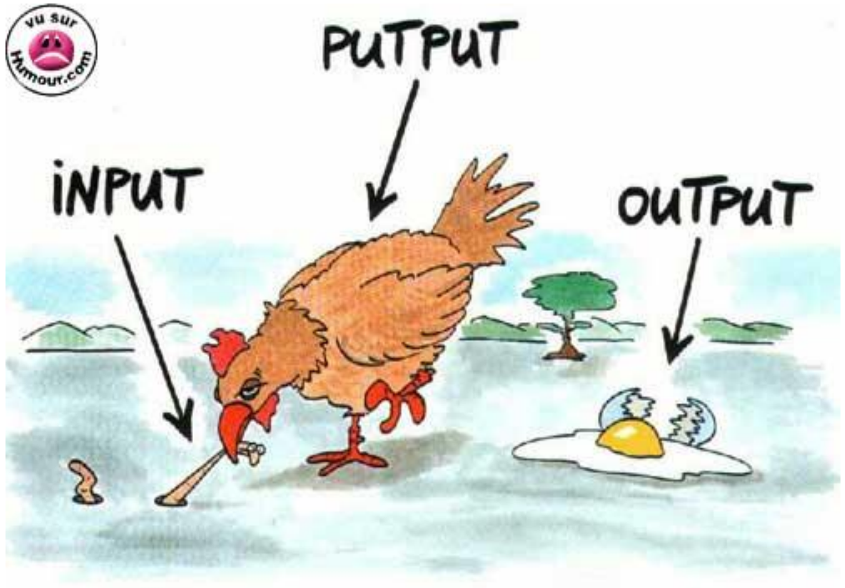
Errors

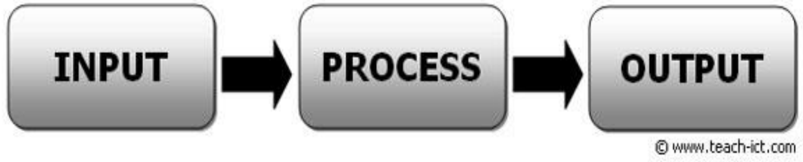
- Syntax Error
- Logical Error
  - ↳ Run Time Error
- Warning

Bug Free Software

Input & Output

- **input/output (I/O)**, refers to the communication between a computer, and the outside world, possibly a human, or another computer. Inputs are data received by the system, and outputs are the data sent from it.





First C Program

```
#include <conio.h>
#include<stdio.h>
```

Directive statements
Ask **preprocessor** to included contents of files conio.h and stdio.h

```
main()
{
  /* print hello world on
      screen */
      printf("hello world \n");
      getch();
}
```

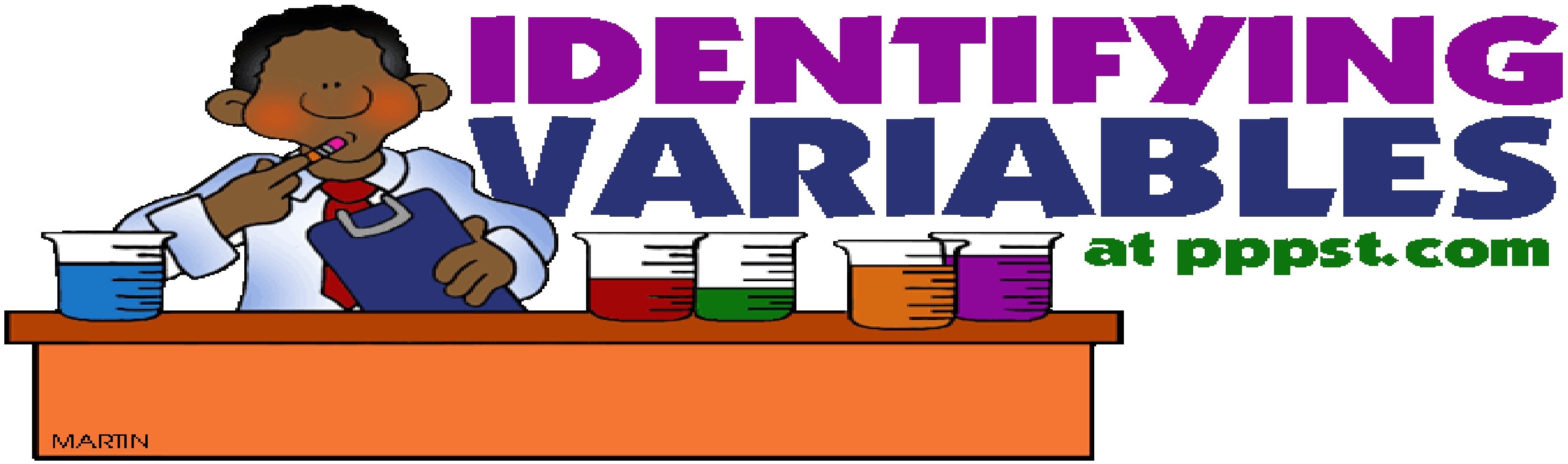
- Program execution begins at main().
- The curly braces {} define the beginning and end of a program block.
- C is case sensitive so main() and Main() are two different function names.
- Statements are terminated with a semi-colon.
- printf() is a standard C function that can be used to output text to the standard consol.
- \n prints a new line.

Storing Data

- Variables
- Memory
- Binary System
- ASCII Table
- Data Types

Variables

- A symbolic name associated with a value and whose associated value may be changed.
- Variable has (value can be changed, data type , location in memory, and name or identifier)
- Variables are declared at the beginning of the function
- How to declare : Data_type identifier;



Identifier

- Is the name of the variable, by which we access the data stored in the variable or the location of the variable in the memory.
- Vary from one to 32 characters
- The first character must be a letter (a to z or A to Z) or an underscore (_). Subsequent characters must be either letters, digits, or underscores
- Identifier names are case sensitive. Num is not as num.
- Identifier name should not have the same name as functions that are in C standard libraries.
- Camel and Pascal casing
- Name should be descriptive

Identifiers

Valid

- size_No1
- go4it
- screenWidth
- PageNumber (pascal)
- backColor (camel).
- _very_nice
- _myPtr

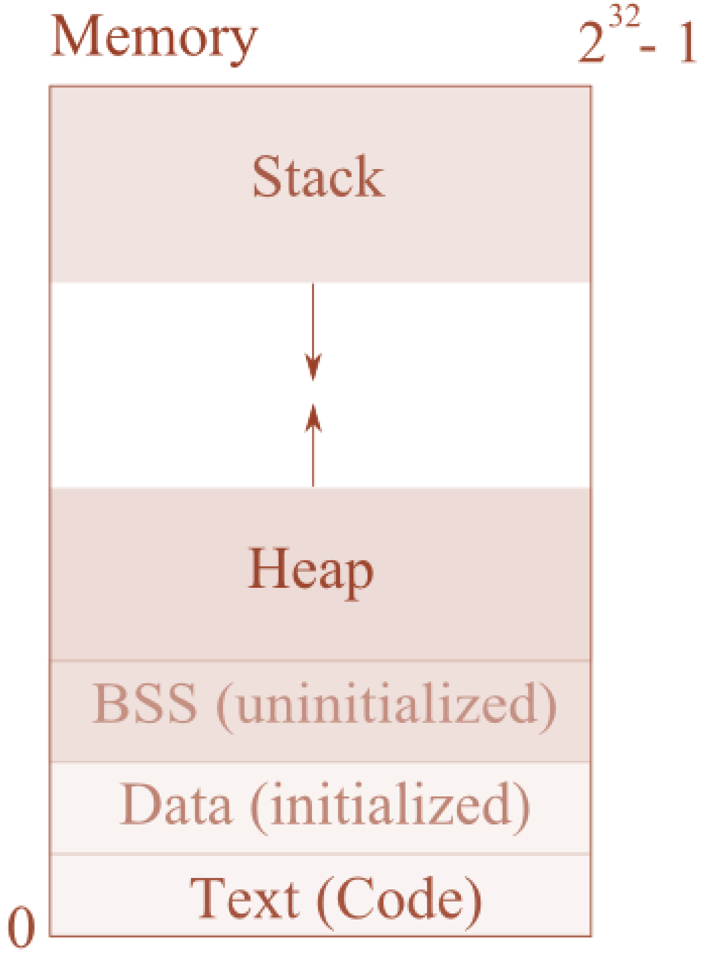
Invalid

- won@last
- Page#8
- $amount
- 3_times
- char
- First name

Memory

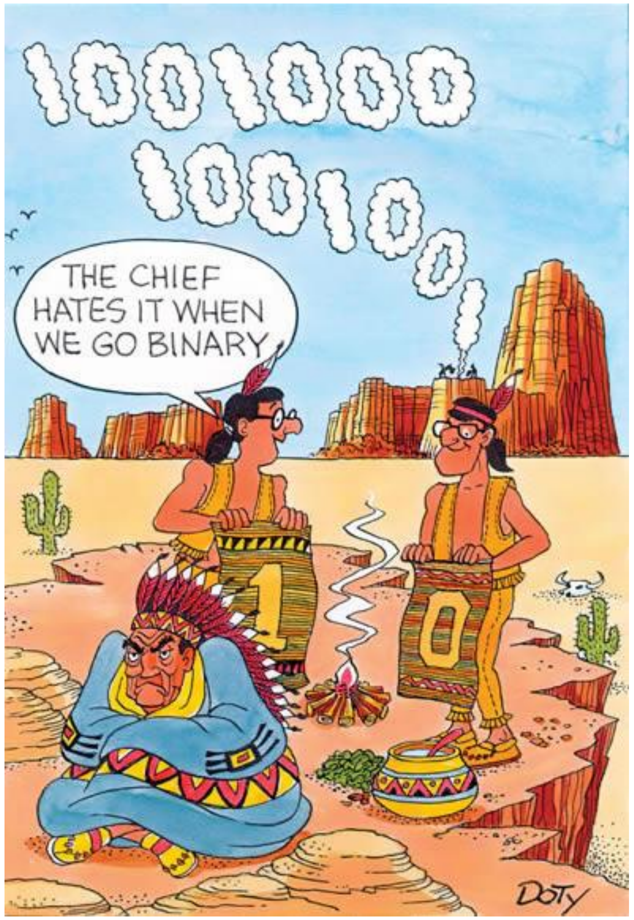
- **Memory** refers to the physical devices (Hardware) used to store programs (sequences of instructions) or data on a temporary (RAM) or permanent (Hard Disk) basis for use in a computer.

Memory – Cont.

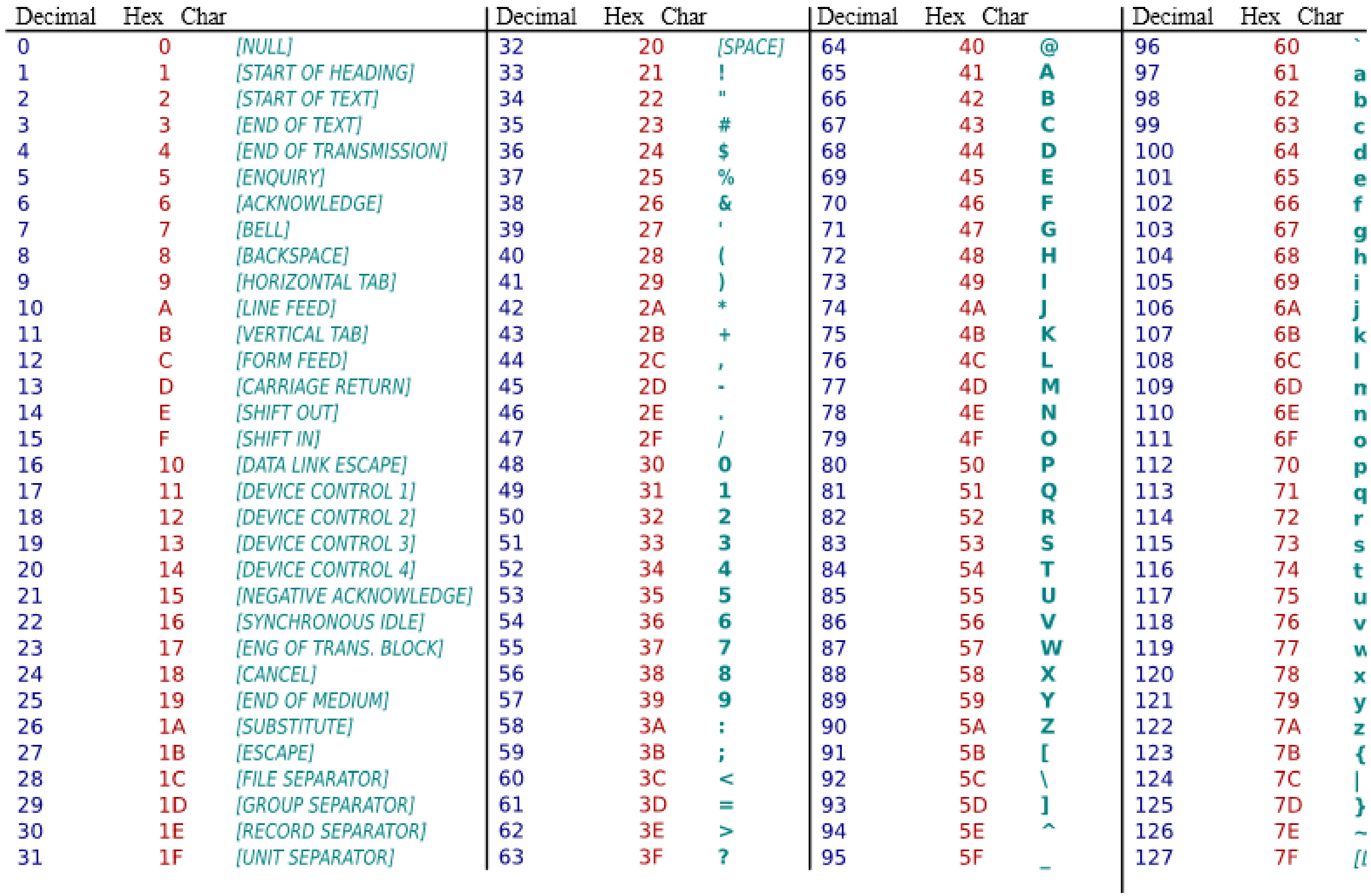

Memory
$2^{32}-1$
Stack
Heap
BSS (uninitialized)
Data (initialized)
Text (Code)
0

Binary System

- To translate from our language to the hardware or machine language they introduce binary language
- Numbers binary
- Character binary (ASCII table)
- Using compiler or interpreter to translate from high level language to binary language



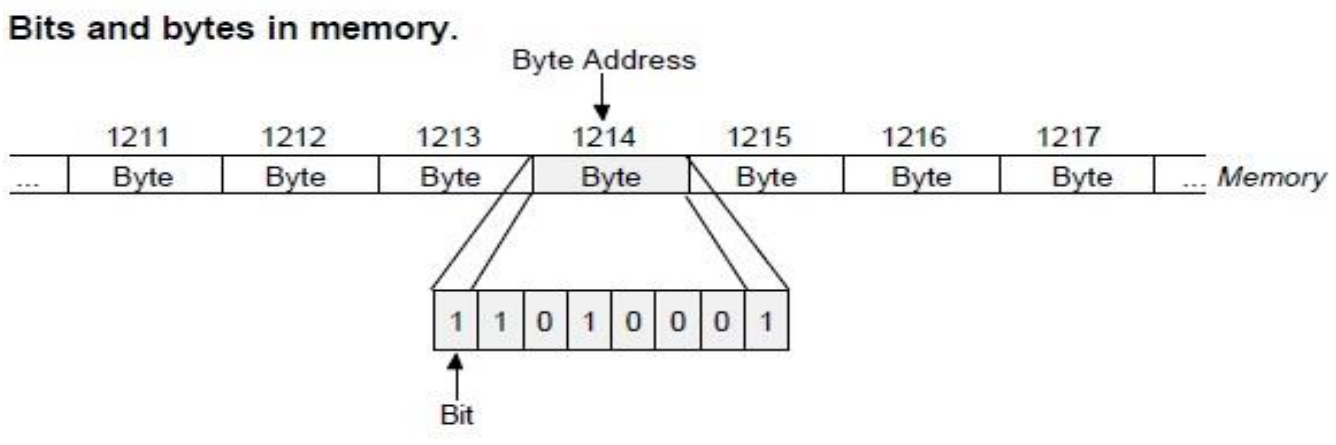
ASCII Table

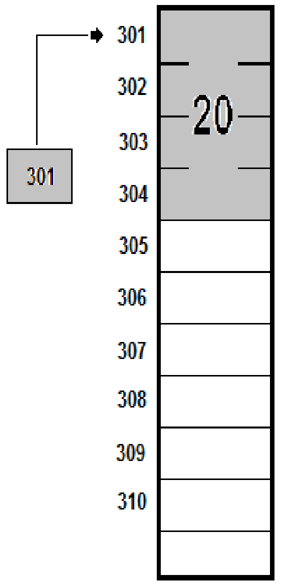


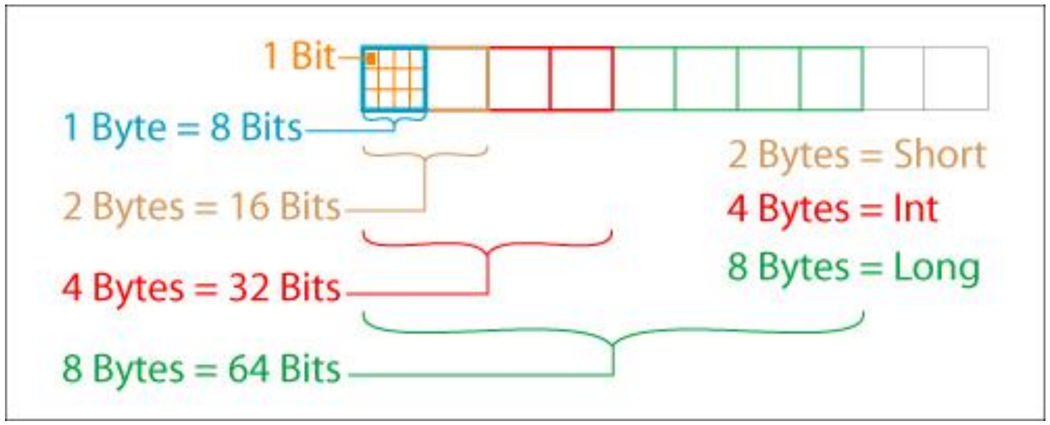
Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[L

Data Types

- Name, Value, Address, Type







Data Types

Type Keyword	Data Type Name	Size in Bits	Minimal Range	Declaration Example
char	Character	8	-128 to 127	char ch;
int	Integer	16	-32,768 to 32,767	int num;
float	Floating-Point	32	Six digits of precision	float f;
double	Double Floating-Point	64	Ten digits of precision	double d;

printf function

Printf is a function in C used to output data to user on screen (standard output) for example:

printf("student age is %d\n", age); which will produce this output:

student age is 33

The first argument of the **printf** function is called the **control string.** When the **printf** is executed, it starts printing the text in the control string until it encounters a %character. The%sign is a special character in C and marks the beginning of a **format specifier.** A format specifier controls how the value of a variable will be displayed on the screen. When a format specifier is found, **printf** looks up the next argument (in this case **age**), displays its value and continues on. The **d** character that follows the % indicates that a (d)ecimal integer will be displayed. At the end of the control statement, **printf** reads the special character \n which indicates print the new line character.

printf function

General syntax of the **printf** function

printf(control string , argument list);

where the ***control string*** can contains

1) literal text to be displayed,
2) format specifiers, and
3) special characters.

The arguments can be variables, constants, expressions, or function calls --anything that produces a value which can be displayed. **Number of arguments must match the number of format specifier** . Unpredictable results if argument type does not "match" the identifier.

Format Specifiers

Type	Format Specifiers
character	%c
Decimal integer	%d
octal	%o
hexadecimal	%x
Long int	%ld
Floating point	%f
double	%lf
string	%s

code	output
int a=66; printf("%d\n",a);	66
int z=66; printf("%c\n",z);	B
int z=66; printf("%X\n",z);	42
int z=66; printf("%o\n",z);	102
float z=66; printf("%f \n",z);	66.000000

printf examples

code	output
printf("AB C\tz");	ABC z
printf("%d\n",6);	6 (cursor goes to next line)
printf("%c %d",'A','A');	A 65

printf("hello");
printf("world");
	hello world

printf("hello \n");
printf("world");
	hello
	world

printf("%x",14);	e
printf("sum = %d",5+6);	Sum=11

scanf function

Scanf is a function in C which allows the program to accept input from the keyboard. As example.

```
main()
{
            int age;
            printf("Please enter in your age \n");
            scanf("%d",&age);
            printf("\n Your age is %d\n",age);
}
```

Output

```
Please enter in your age
25
Your age is 25
```

1- An integer variable called **age is** defined.
2-A prompt to enter in a number is then printed with the first **printf** statement.
3-The **scanf** routine, which accepts the , has a **control string** and an **address list**. In the control string, the format specifier **%d** shows what data type is expected. The **&age** argument specifies the memory location of the variable the input will be placed in. The **& character is the address operator.**
4-the entered data is confirmed with the second **printf** statement.

scanf function

You can accept more than one value with the same scanf

```
main()
{
        int x,y;
        printf("Please enter  the coordinates \n");
        scanf("%d%d",&x,&y);
        printf("\n your location at  x=%d and y=%d \n",x,y);
}
```

Output

```
Please enter  the coordinates
25 33
Your location at x= 25 and y=33
```

Some of used Functions

- getch() : conio.h

Gets a character from user.

e.g., n=getch();

- clrscr() : conio.h

Clears the screen.

- gotoxy(x,y) : conio.h

Move the cursor position.

Assignments in Lab

- **Some simple assignments to cover in lab**
  - A program to display Hello World.
  - A program to take a character from the user, and then display its ASCII code.
  - Same program but vice versa.
  - Program to take an integer (decimal) and display its hexadecimal equivalent.
  - A program to take two numbers and print the sum, subtraction, multiplication.

Thank You!