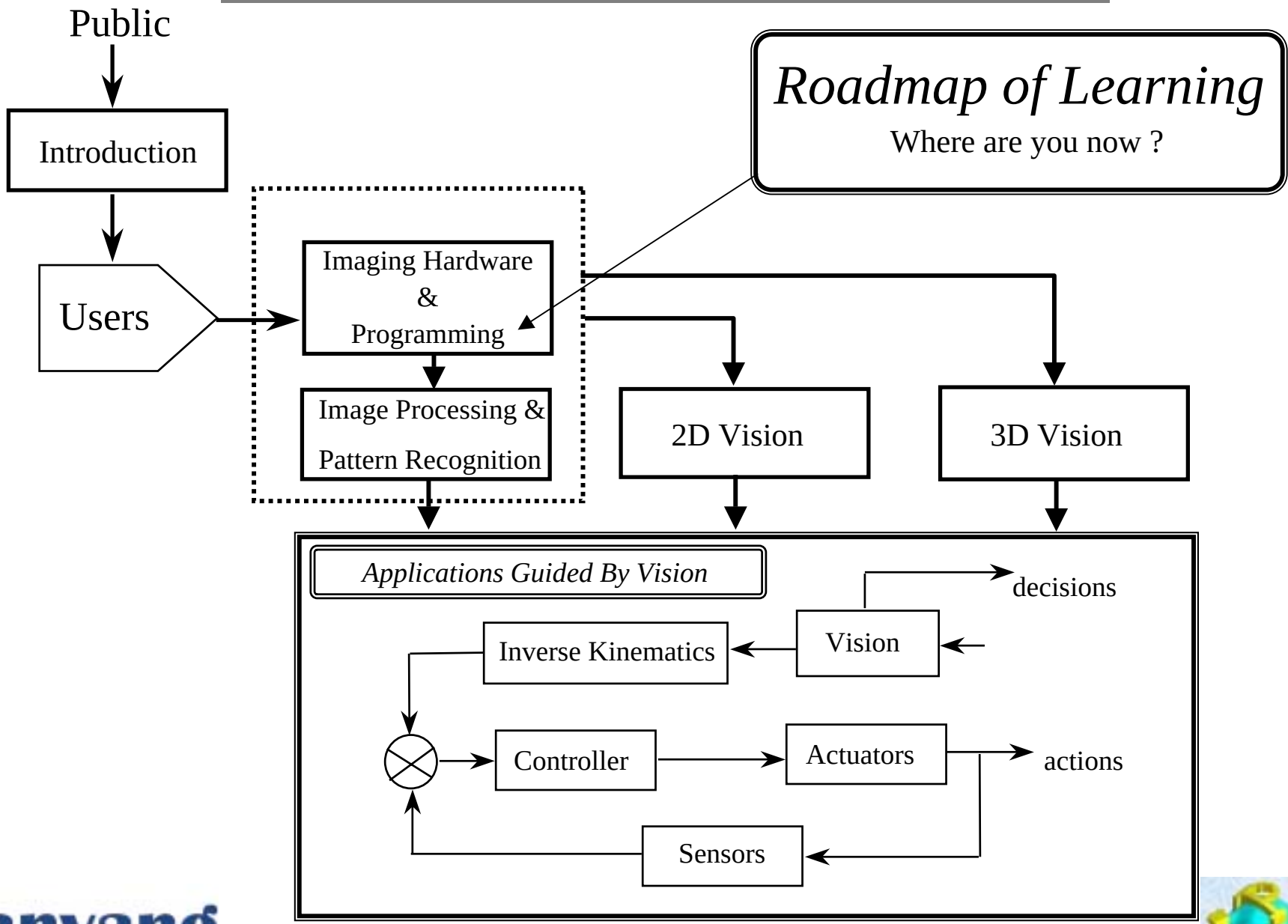
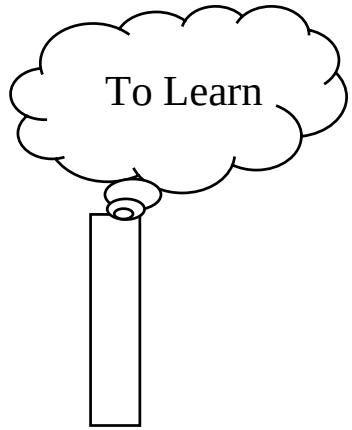


CONTENT



Chapter 3: Fundamentals of Programming

3.1 Basics of C programming

3.2 Data Structures in C

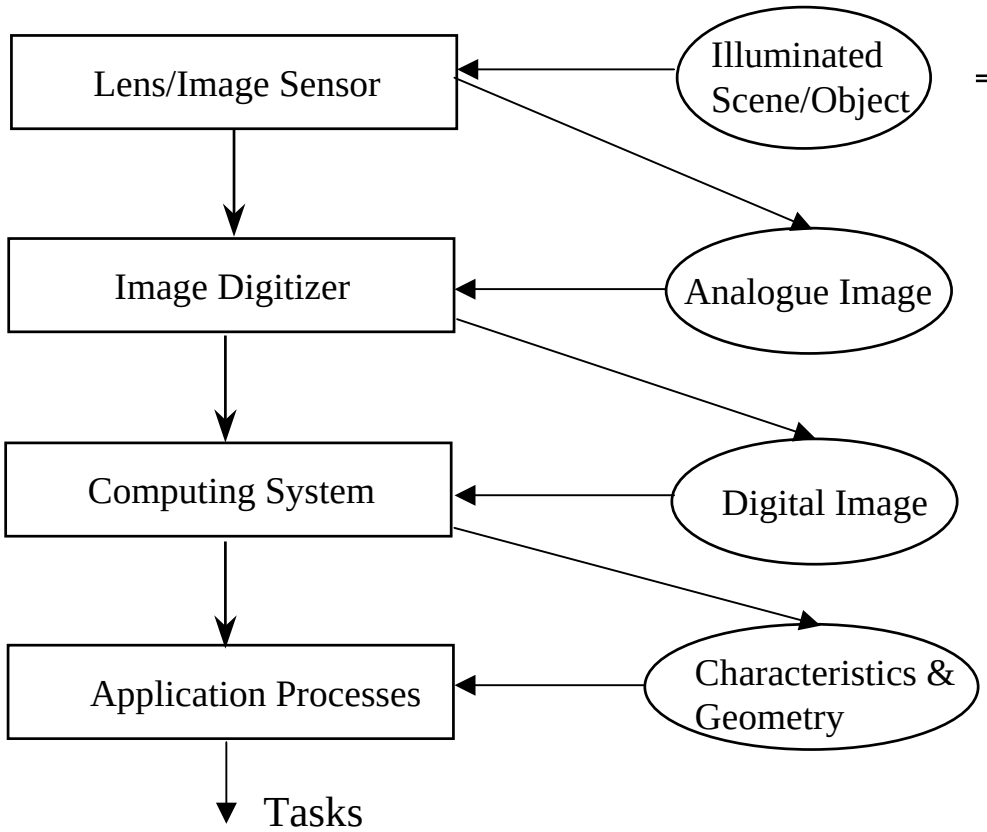
3.3 Data Processing Loops in C

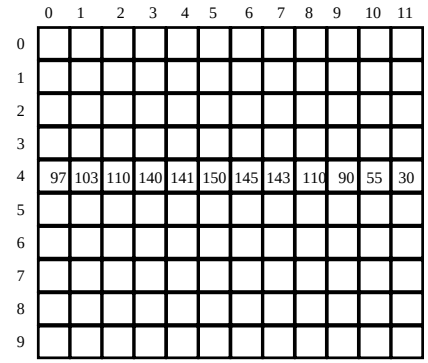
3.4 Use of Image and Graphics Library

3.5 Memory Concept

What is a machine vision system ? (A Review)

ANSWER:

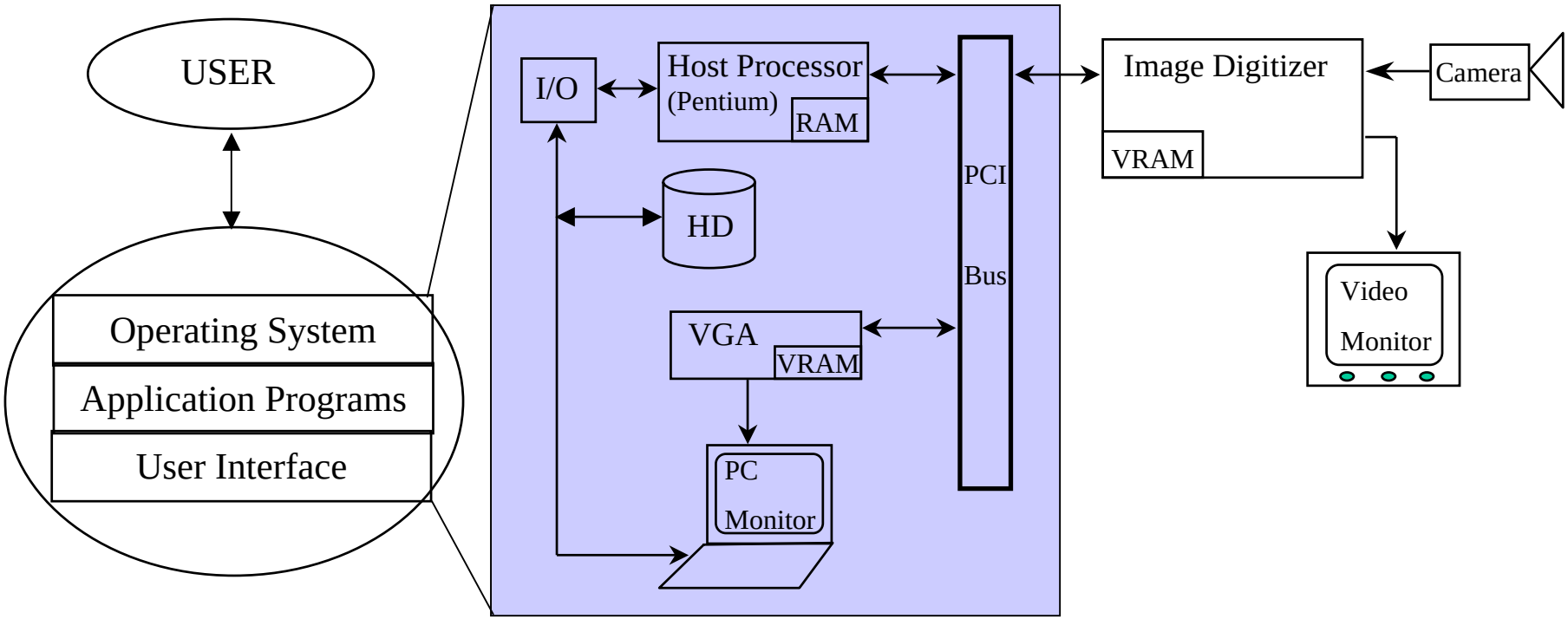




Question:

How to program a machine vision application ?

A computing system is a digital system. At the lowest level inside a computer, what can it understand ?



ANSWER:

1. Human language ? (YES or NO ?)
2. Programming language ? (YES or NO ?)
3. Binary codes (or machine codes) (YES or NO ?)

Is it convenient for human users to dialogue with a computer by using machine codes like 01001111001111 … ?

ANSWER:

(YES or NO ?)

What is the most acceptable solution for human-computer dialogue ?

ANSWER:

To invent and use languages that are readable by both human and computer, ie., the programming languages (C, C++, others …) !

What is a programming language ?

ANSWER:

It is a system that has the following major components:

* Constants and variables (--> words)

* Types (--> word classes: verb, noun, adjective …)

* Syntax (--> grammatical rules)

* Instructions/expressions (--> sentences)

* Functions (--> chapters)

* Comments (--> footnotes)

What is the output of a programming language ?

ANSWER:

Program

(It is a collection of constants, variables, expressions, instructions, functions, and comments) (--> document).

How is a program stored inside a computing system ?

ANSWER:

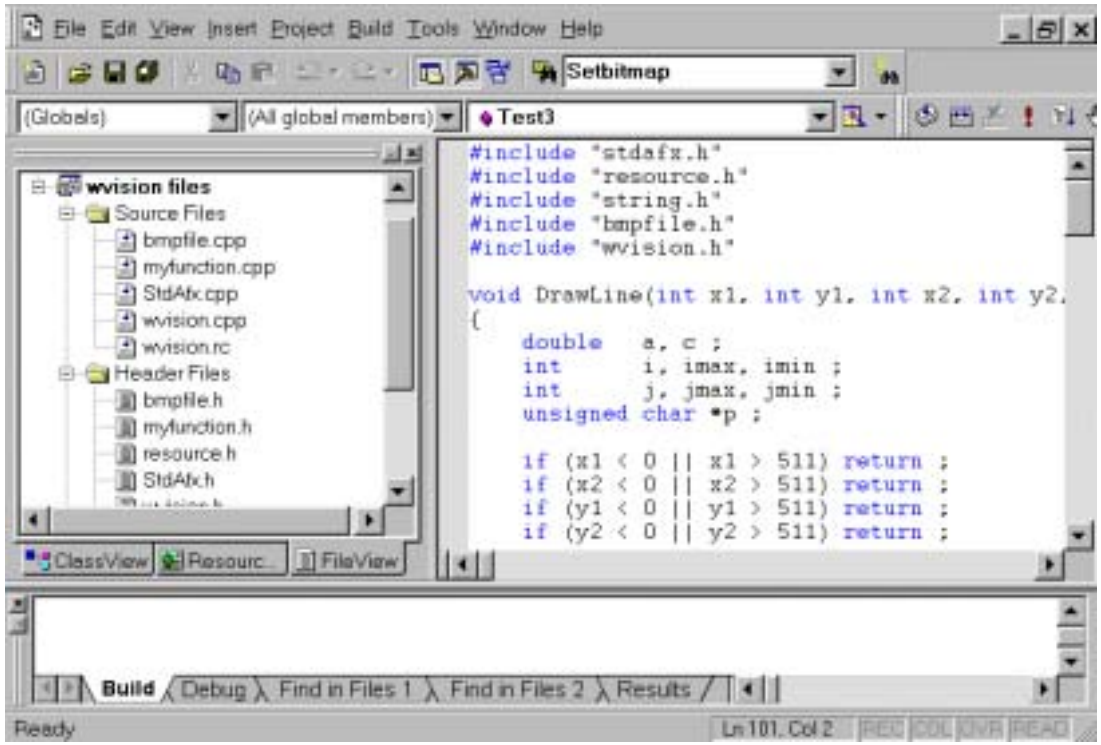
ASCII File on Hard Disk (It is a storage media that contains a series of ASCII codes).

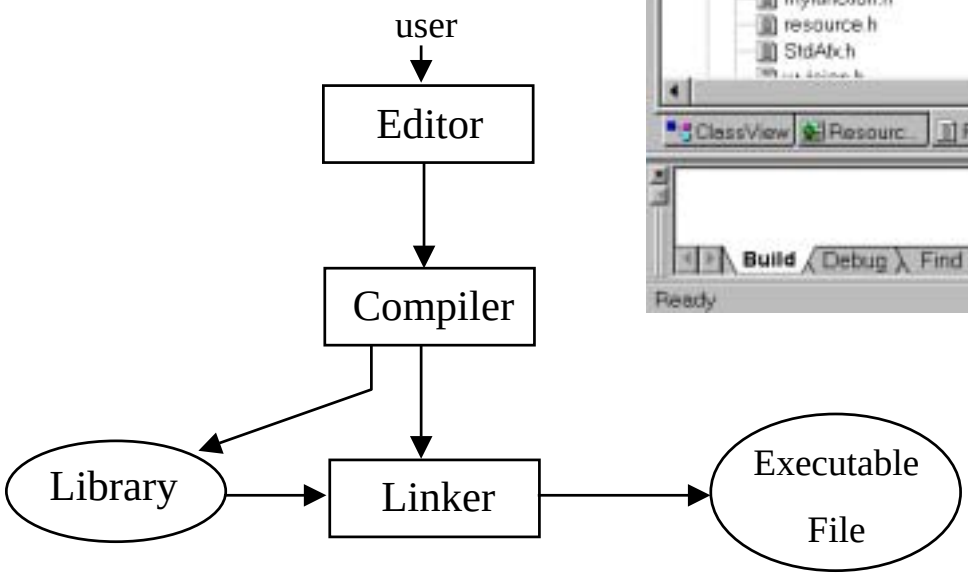
ASCII stands for “American Standard Code for Information Interchange”

How to develop an executable program ?

ANSWER:

- Editor
- Compiler
- Linker
- Loader
- Debugger





How to define constants in C ?

Examples:

```
#define   WIDTH    512
#define   HEIGHT  512
```

```
enum Color  {Red, Green, Blue} ;

void  foo()
{
    Color   color_of_my_cloth;
    color_of_my_cloth = Red ;
}
```

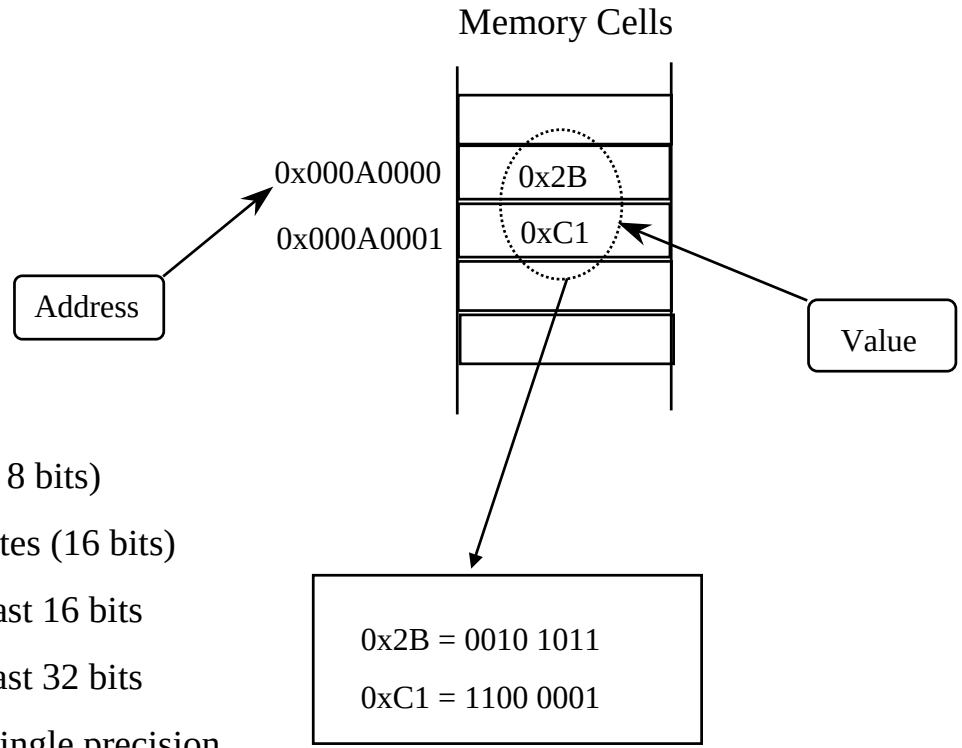
How to define and create a variable in C ?

ANSWER:

Syntax: “<type> <name of variable> ; “

```
int         x, y ;
double   z ;
```


What are the common types in C ?



ANSWER:

char	:	character (1 Byte; 8 bits)
short	:	an integer of 2 Bytes (16 bits)
int	:	an integer of at least 16 bits
long	:	an integer of at least 32 bits
float	:	a real number of single precision
double	:	a real number of double precision

(the qualifier “unsigned” can be applied for an integer)

How to define and create a function in C ?

ANSWER:

1. A function is a collection of related instructions/expressions.

2. A function can produce a result as returned value.

```
void PrintName()
{
    printf("\n> my name is XM") ;
}
```

```
int GetStudentNumber()
{
    int     nb_student ;
          …...
    return nb_student ;
}
```


How to assign value to a variable in C ?

ANSWER:

a) directly

```
void foo()
{
    int    x, y ;
    x = 2 ;   y = 10 ;
}
```

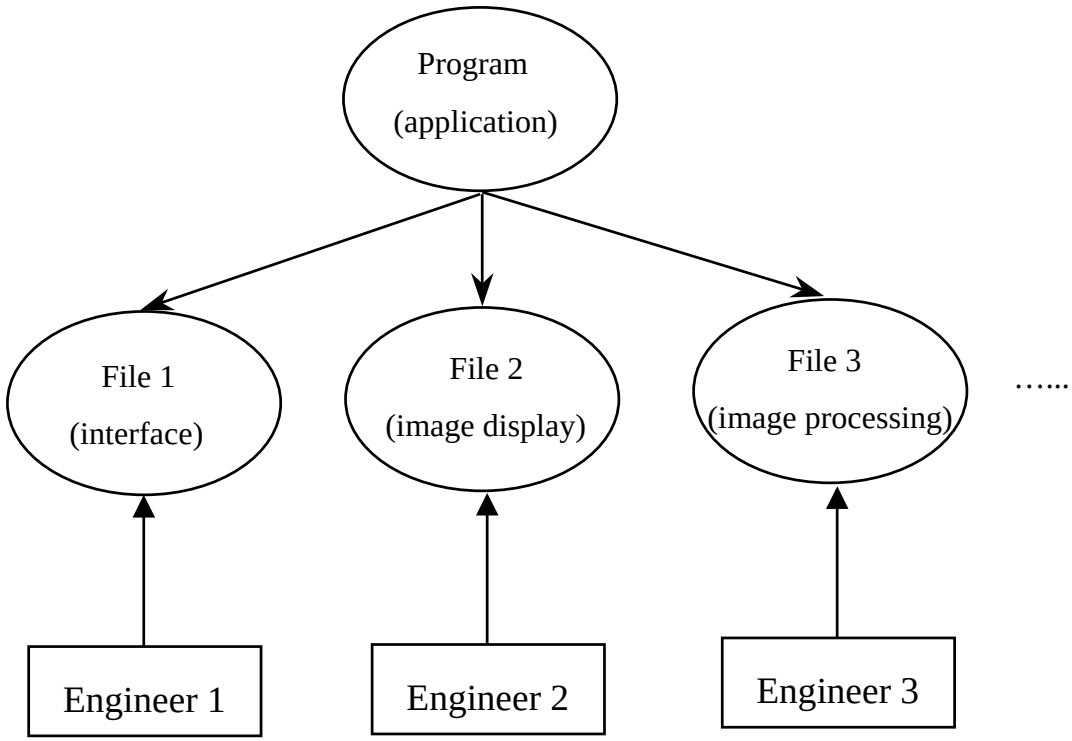
b) by arithmetic expression

```
void foo()
{
    int    x, y, z ;
    x = 2 ;   y = 10 ;
    z = (x*x + y*y) ;
}
```

c) by a function call

```
void foo()
{
    int    x, y, z ;
    x = 2 ;   y = 10 ;
    z = sqrt(x*x + y*y) ;
}
```


If a program is very long, it is wise to functionally divide it into a set of files:



Problem:

How to handle the visibility of variables and functions across the files ?

How to handle the visibility of variables and functions across the files in C ?

ANSWER:

To make use of the "visibility" constraints in C:

* static : visible within a file
* (local) : visible within a function
* (global) : visible across all files

File 1

```
int   x,  y ;
static  int  z ;

void f1()
{
    double   a ;
    a = 1.3 ;   z = 8 ; x=3;
}
```

File 2

```
extern int   x,  y ;
static  int  z ;
double     n ;

static void f2()
{
    n = 1.5 ; y = 2 ; z = 9;
}
```

File 3

```
extern int          x,  y ;
extern double     n ;

void f3()
{
    f1() ;
    n = 1.2 ; x = 10 ; y = 1;
}
```


In order to prevent data interference or conflict, it is important to enforce the use of visibility constraints. The following are two rules:

Rule 1: By default, always declare the variables and functions as "static" (to make them only visible within a file)

```
static int      x,   y ;

static   void   InitApplication()
{
    x = 0 ;   y = 0 ;
}

void   main()
{
    InitApplication() ;
}
```


Rule 2: If a variable or function has to be global, make it global and explicitly indicate this situation in a conjugate header file *.h.

toto.c

```
char lecture_name[100];

int GetStudentNumber()
{
    int   nb_student ;

    return nb_student ;
}


main()
{
}
```

toto.h

```
extern char lecture_name[100];

extern int GetStudentNumber() ;
```


Example

To print the name of the lecturer for a given subject:

whoislecturer.c

```
#include “printname.h”

int     subject_code ;

main(int argc, char **argv)
{
    subject_code = 483 ;
    PrintName() ;

    subject_code = 432 ;
    PrintName() ;

}
```

whoislecturer.h

```
extern int     subject_code ;
```


printname.c

```
#include   <stdio.h>
#include   "whoislecturer.h"


void  PrintName()
{
    if (subject_code == 483)
        printf("\n The lecturer is X");
    else
    if (subject_code == 432)
       printf("\n The lecturer is M");
    else
       printf("\n Don't know !") ;
}
```

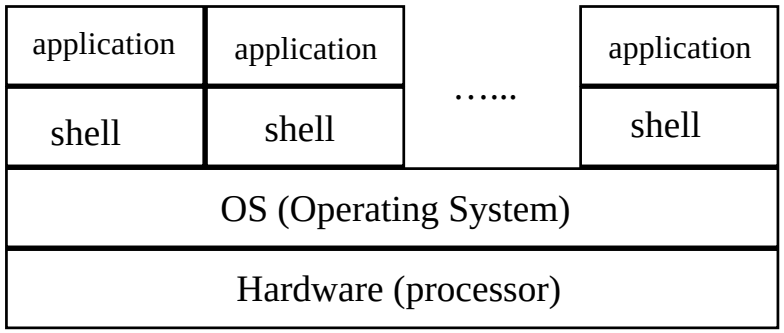
printname.h

```
extern void     PrintName() ;
```

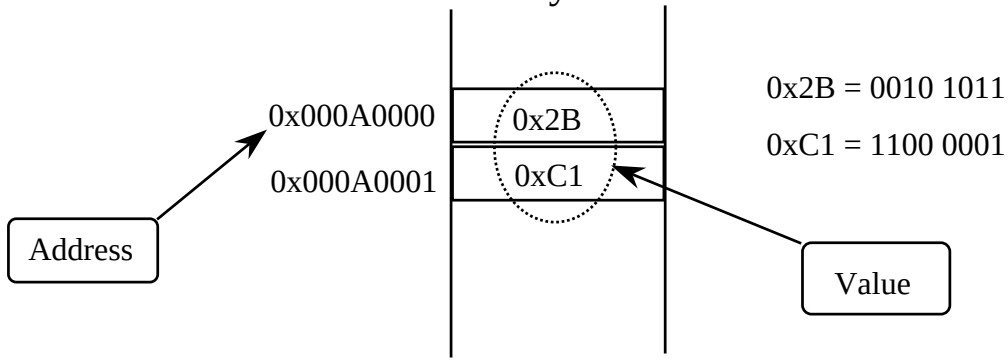

This example tells you four more things:

1. A C-program always starts with the “main()” function !!!

2. A function may have many arguments as input. The “main()” function is always activated by a “shell” program of the operating system:



3. With a defined variable, one can not only access to the value but also access to the memory location where the value is stored !

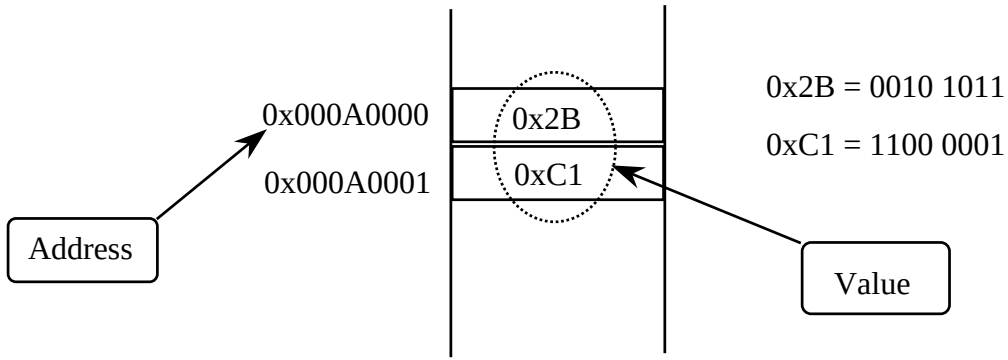



```
void  foo()
{
    int   x  ;
    x = 0x2BC1 ;
}
```

In C, the address of a variable x is accessed by "&x" (in this example, x=0x2BC1 while &x = 0x000A000).

"Can one define a variable to hold an address ?

4. In C, one can define a variable to hold an address. This type of variable is called “pointer”.



In C, one can assign address and value if using a variable of “pointer”. The value of a pointer x is accessed by “*x”.

```
void   foo()
{
   int    *y ;

   y = 0x000A001 ;
   *y = 0x2BC1 ;
}
```


Question:

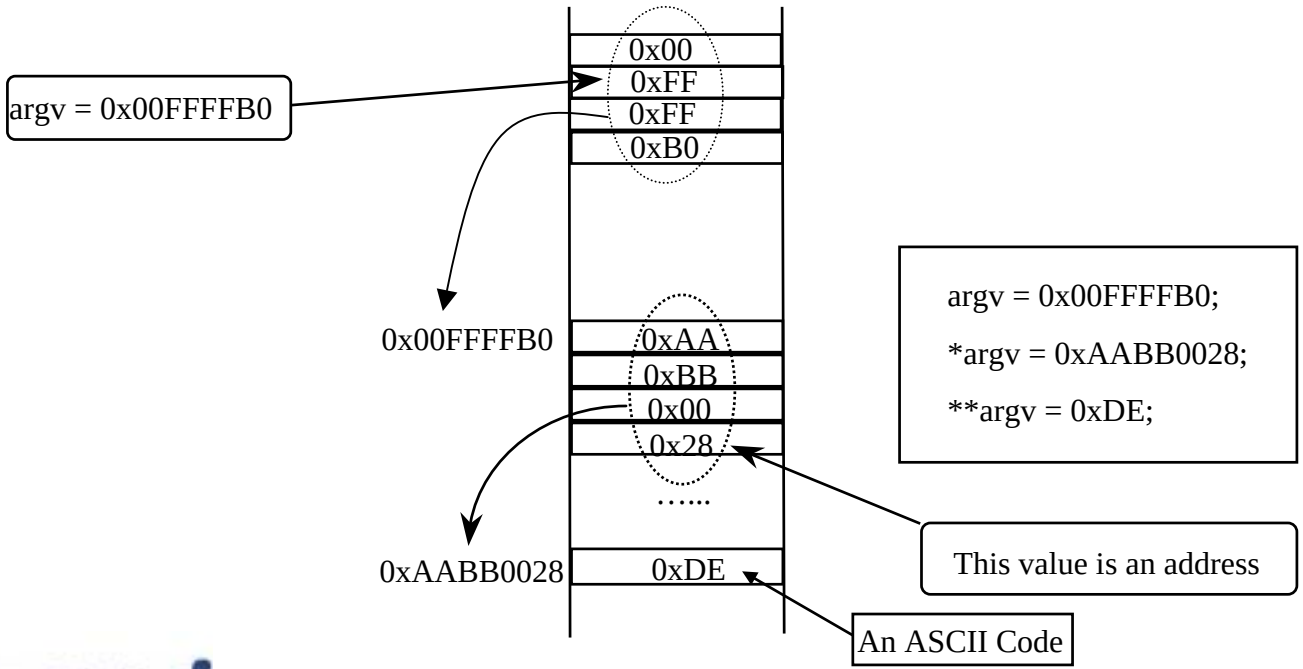
What does the pointer in “main(int argc, char **argv)” mean ?

ANSWER:

argv holds the address of a value

that is also another address

where a character is stored.



How to translate the ASCII coded file(s) of a program into an executable binary file in C ?

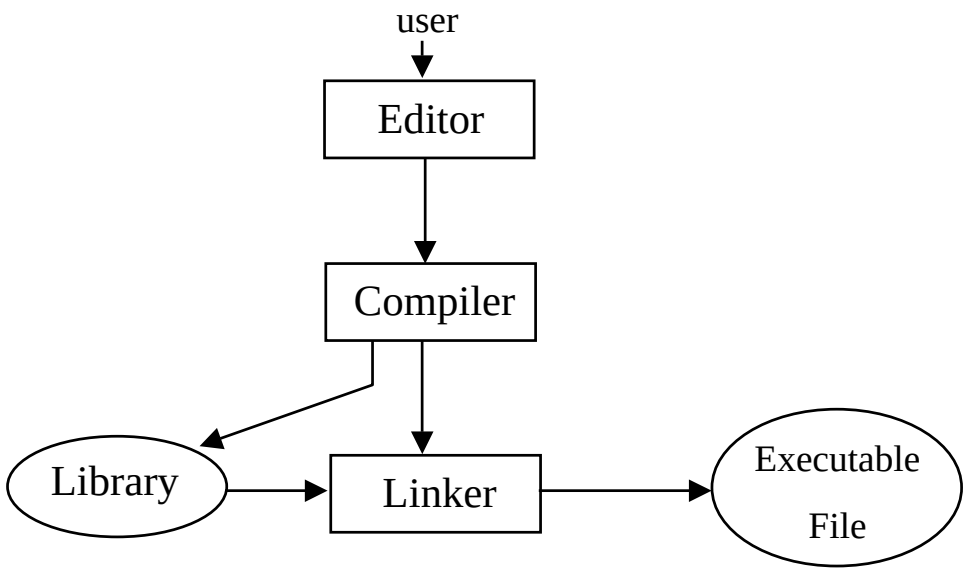
ANSWER:

a) To use Compiler to convert ASCII coded files into “object” files.

(*.c --> *.o)

b) To use Linker to combine the “object’ files and the needed library files into executable file.

(*.o + *.a --> executable file)



Example : toto.c

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

main(int argc, char **argv)
{
    printf("\n The square root of 9 is: %f", sqrt(9.0)) ;
}
```

Step 1: Conversion of ASCII coded program file into "object" file:

INPUT: toto.c

OUTPUT: toto.o

COMMAND: cc -c toto.c

(this will generate toto.o)

Step 2: Linking of object file(s) and library file(s) into an executable file:

INPUT: toto.o and libm.a

OUTPUT: toto

COMMAND: cc -o toto toto.o -lm

("-lm" means to use the library libm.a)

(the result will be an executable file named "toto")

SUMMARY

1. Human and computer dialogue is based on the use of programming languages.

2. A programming language includes:

   * constants, variables and functions
   * types
   * arithmetic operators (+, - *, /, …)
   * visibility constraints
   * compiler
   * “make” utility, etc.

3. For a large program, it is wise to divide it into a set of ASCII files. One must pay attention to visibility conditions of variables and functions across files.

4. It is a good practice to group all the global variables and function inside a program file into a corresponding header file.