BADJI MOKHTAR UNIVERSITY

SCIENCE AND TECHNIQUES DEPARTMENT
1st year ST

Course 4 : C programming language basic concepts

by

Dr. Samira LAGRINI

lagrini.samira83@gmail.com

Course link : https://elearning-facsci.univ-annaba.dz/course/view.php?id=469

What is a program/ programming language ?

- A program is a sequence of instructions that a computer follows to solve a problem

- A programming language is a set of words and symbols and codes that enables human to write a computer program.

```c
#include <stdio.h>
int main() {
    int number1, number2, sum;

    printf("Enter First Number: ");
    scanf("%d", &number1);

    printf("Enter Second Number: ");
    scanf("%d", &number2);

    // calculating sum
    sum = number1 + number2;

    printf("\nAddition of %d and %d is %d", number1, number2, sum);
    return 0;
}
```

C programming language

- C is a programming language initially developed by Dennis Ritchie in 1972
- C is a compiled language (as opposed to interpreted languages).
- In compiled languages, the entire program is translated into machine code at once.
- In interpreted languages, the program is translated and executed line by line.
- To program with a compiled language such as C, it is necessary to first install the compiler for that language,

- *Example of C compiler* : ***Turbo C***, *Turbo C++, Dev c++, Borland C...*

General Structure of a C program

```
# include<stdio.h>
main()
{
    < déclaration des variables>
     < Instructions;>

}
```

Example:

```
#include <stdio.h>
   main()
{  float x, Y;
   printf("entrez un nombre réel ");
   scanf("%f", &x);
   Y=2*x;
   printf( "son double  = %f\n",Y);
}
```

Basic elements of a C program

a C program consists generally of the following elements:

- Identifiers
- The key words
- The constants
- The variables
- Input/output functions
- Comments
- …

1. Identifiers

- Identifiers refers to name given to entities such as variables, functions, structures etc.
- Identifiers are created to identify an entity during the execution of the program.
- Identifier names must be different from keywords

❖ Rules for naming identifiers

1. A valid identifier can have letters (both uppercase and lowercase letters), digits and underscores (_).
2. The first letter of an identifier should be either a letter or an underscore, but not a degit
3. Identifiers are case-sensitive: x1 and X1 are considered 2 different identifiers,

2. The Keywords

- Keywords are reserved words used in programming that have special meanings to the compiler.
- Keywords are part of the syntax and they cannot be used as an identifier
- In C, there are 32 keywords:
- *double- float- int- short- struct- unsigned-break- continue -else for- long- signed- switch- void-case- default- enum – goto - register- sizeof- typedef- volatile-char- do- extern- if- return- static –union- while- Auto- const.*

3. The variables

- A **variable** is a memory location used to store a specific value during the execution of a program.

A **name** is assigned to the variable to uniquely identify it among others.

- Syntax:

Type name-variable ;

- ❑ A variable must be declared before its use.
- ❑ When a new value is assigned to a variable that already holds a value, the existing value is replaced by the new one.

Example

```
int x;
float y=23,5,
```

Predefined types in C

- The type of an object (variable, constant or function) defines how it is represented in the memory.
- It allows us to specify the range of values that the variable can take as well as the operations that can be performed with it.
- The basic types in C are:

Type	its meaning
_bool	an integer that can take two values: 0 or 1
int	an integer
short	
long	
unsigned	An unsigned integer
char	a character
float	Floating point real numbers. They correspond to the different possible precisions.
double	
Long double	

4. The Constants

- A constant is an object containing a value that can never be changed.
- Syntax:

```
# define  name  value
```

Example :

```
# define X 100
```

Example: Calculate perimeter of a circle

```
# include <stdio.h>
# Define pi 3,14      /* declaration of a constant pi */
  main()
 {
  float R1, P; /* declaration of two variables*/

   printf("enter the length of the radius"); /* displaying a message */

   scanf("%f", &R1); /* reading the variable R1*/

   P=2*pi*R1; /* calculate the perimeter of a circle */

   printf( "  the perimeter of the circle = %f\n", P);
 }
```

C Input/ Output (I/O)

5. C Output

- printf() is one of the main output function. The function sends formatted output to the screen.

- The printf function prints:
  - **A string inside quotations**: Example: printf ("**hollo**")
  - **A value in a specified format**. The syntax for this is:

printf ("format specifier " , x);

Example:

```
printf ( " mon programme en C");
printf ( "la surface= %f " , X) ;
```

5. C Output ‘printf ()’

- Format specifiers are **(%d, %f, %c, %s)**
- They designate the printing format.

Format	Conversion to
%d	int
%f	float
%c	char

C Output ‘printf ()’

Escape Sequence in C

sequence	meaning
\n	New line
\t	horizontal tab
\v	Vertical tab
\r	a return to the start of the current line.
\\	The character \

Example :

```
#include <stdio.h>
main()
{
int i =23;
char c = 'A';
Printf (" print of i: \n");
printf("%d \n ", i);
Printf (" print of C: \n");
printf("%c \t  %d ", c, c);
}
```

This program prints on the screen:

```
print of i:
23
print of C:
A     65
```

6. C input ‘ scanf ’

- scanf() is one of the commonly used function to read formatted input from the keyboards.
- The syntax of the scanf() statement is as follows:

scanf(" *format specifier*", &*X*);

- As for printf, the *format specifier* can be (%d, %f,% c ,% s)

Example

```
#include <stdio.h>
main()
{  float x;
   printf("enter a number  x = ");
   scanf("%f", &x);
   printf("x = %f\t",x);
 }
```

7. Comments

- Comments in a program are messages that explain parts of the source code.
- comments can be placed anywhere in the program.
- A comment is written either between /* and */ or after two slashes

Example

```
/* This is a comment that can be
      heard on several lines */
// This is another comment
```

Comments

```c
#include <stdio.h>  // Include the standard input/output header

int main() {
    int age;  // Declare an integer variable to store the user's age

    // Print a prompt to the user asking for their age
    printf("Enter your age: ");

    // Read the user input and store it in the 'age' variable
    scanf("%d", &age);

    // Print the entered age
    printf("Your age is: %d\n", age);

    return 0;  // Return 0 to indicate the program ended successfully
}
```