Chapter 4 :

C programming language

Table of contents

OVERVIEW

C programming is a widely used and influential programming language that was developed in the early 1970s by Dennis Ritchie at Bell Labs. It is a high-level, general-purpose programming language known for its flexibility, efficiency, and low-level programming capabilities.

C programming has had a significant impact on the development of modern programming languages and systems. It's commonly used in areas like operating system development, device drivers, embedded systems, where fine-grained control over hardware and resources is essential.

C program

The general structure of a C program consists of various components and follows a specific format.

Header	**Include standard C library header**
Declaration	**Global Declarations/Define global variables and functions**
Instructions	**Main Function** **Local Declarations/ Define local variables** **----------** **The instructions/Main body of the program.** **----------.** **output statement** **END main function.**

C Comments

- In C, you can insert text as comments to provide explanations, notes, or documentation within your code. Comments are ignored by the C compiler and are intended solely for human readers.

- **Single-Line Comments:** Single-line comments are used for brief comments on a single line. You can start a single-line comment using //, and it continues until the end of the line.

```
// This is a single-line comment
```

- **Multi-Line Comments:** Multi-line comments are used for longer comments that span multiple lines. You begin a multi-line comment with /* and end it with */.

```
/* This is a multi-line comment
It can span multiple lines
Useful for longer explanations */
```

C library

The C Standard Library is a set of functions, macros, and constants that provide core functionality for the C programming language.

- **<stdio.h>**: Input and output operations. Functions like printf, scanf, fopen, and others are part of this header.
- **<stdlib.h>**: General utilities. Functions such as malloc, free, exit, and others for dynamic memory allocation, random number generation, and program termination.
- **<string.h>**: chain of characters or text manipulation functions like strcpy, strlen, strcmp, etc.
- **<math.h>**: Mathematical functions, such as sqrt, sin, cos, and others.

C variables declaration

- In C, you declare variables with a specific data type to specify the type of data and name that variable can store.

Example for

Integer Variables (int):

```
int age;
int x, y, z; // You can declare multiple int variables in one line.
```

Floating-Point Variables (float or double):

```
float price;
double pi = 3.14159; /* Initialization is optional but can be done at declaration.*/
```

Character Variables (char):

```
char grade;
char symbol = '$'; /*You can initialize char variables at the time of declaration.*/
```

Assignment

signe	utilisation	équivalent
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
<<=	x <<= y	x = x << y
>>=	x >>= y	x = x >> y
&=	x &= y	x = x & y
^=	x ^= y	x = x ^ y
|=	x |= y	x = x | y
++	x++	x = x +1
--	x--	x = x — 1

Input statement(scanf)

The scanf function is a standard input function in the C programming language, and it is used to read formatted data from the standard input (usually the keyboard) into variables.

It is part of the standard input/output library (stdio.h).

scanf("%f", &Variable);

%d: Reads an integer.
%f: Reads a floating-point number.
%lf: Reads a double-precision floating-point number.
%c: Reads a character.
%s: Reads a string (sequence of characters excluding whitespaces).

C Output statement (printf)

The printf function in C is used for formatted output. It allows you to print data to the standard output (usually the console) with a specified format. **printf(" la Somme= %d",S);**

Les types utilisables sont les suivants :

	Type de donnée à afficher	**Caractère de formatage**
Numériques	Entier décimal signé	d
	Entier décimal non signé	u ou i
	Entier octal non signé	o
	Entier hexadécimal non signé	x (les caractères 'a' à 'f') ou X (les caractères 'A' à 'F')
	Flottants de type double	f, e, g, E ou G
Caractères	Caractère isolé	c
	Chaîne de caractères	s
Pointeurs	Pointeur	p

My First c program

```
//my first c program
/* Program square */

#include <stdio.h>
float number, square;
 int main() {
        printf("Enter a number: ");
        scanf("%f", &number);
        square = number * number;
        printf("The square of %lf is %lf\n", number, square);
 return 0; }.
```

Conditional statements (if..else)

- **algorithm code** **C code**

```
if (condition) begin_if
    instructions
end_if
```

```
if (condition) {
    instructions ;
}
```

```
if (condition) begin_if
    instructions 1
end_if

else begin_else
    instructions2
end_else
```

```
if (condition ) {
instructions1;
}

else {
instructions2;
}
```

Conditional statements (if..else) (2)

Example

Write a program that asks the user to enter two integers. The program then displays the larger of the two.

C code

```
#include   <stdio.h>
int    A,B;
int main ()
 {
  printf("donnez le premier nombre");
 scanf(" %d  ", &A);
 printf("donnez le deuxième nombre");
 scanf(" %d  ", &B);
 if  (A > B)  {
 printf(" %d  est supérieur à  %d", A,B);
         }
 else  {
printf(" %d est supérieur à  %d", B,A);
          }
 return 0; }
```

switch statement

- Pour éviter les imbrications d'instructions if, le C possède une instruction qui crée une table de branchement : c'est l'instruction switch
- **La syntaxe du switch est résumée:**

```
switch   (Variable)    {
        case value1 :   <bloc d'instructions 1> ;
                        break ;
        case value2 :    <bloc d'instructions 2> ;
                        break ;
        case valueN:    <bloc d'instructions 3> ;
                        break ;
        default :       <bloc d'instructions 4> ;
                        break ;
                        }
```

switch statement

Example (basic calculator)

```
#include   <stdio.h>
char c;   float  a,b,res;
int main () {
      scanf("%f",&a) ;
      scanf("%f",&b) ;
      scanf("%c",&c) ;
       switch(c) {    case '+':  res=a+b;
                                 break;
                      case '-':  res=a-b;
                                 break;
                       case '*':  res=a*b;
                                  break;
                       case '/':  if (b!=0)    res=a/b;
                                  else printf("erreur de division par zero");
                                   break;
                       default: printf(" cette opération est erronées");
                    }
      printf("Le Résultat = %f",res);
      return 0 ;}
```

Iteration statement/Loop

algorithm code	C code
for (counter=star to stop with step) **begin_for** instructions **end_for**	**for**(counter=star; stop; counter=counter+step) **{** instructions; **}**
while (condition) **begin_while** instructions **end_while**	**While (**condition) **{** instructions **; }**
Repeat begin_repeat instructions **end_repeat** **while (Condition)**	**do {** instructions; **}** **While (Condition);**

Iteration statement/Loop (1)

Example (for loop)

Write a C program that calculates: $S = \sum_{i=1}^{N} \frac{i+1}{-(2i+1)^2}$

```
#include   <stdio.h>

#include   <math.h>
        int  N,i; float  S;
        int main ()
       {    printf("donner un nombre  N:");
            scanf("%d",&N) ;
           S=0 ;
           for (i=1; i<=N ; i++)
           {
            S=S+ (i+1)/(-pow(2*i+1,2));
           }
           printf("La somme = %f ",S) ;
           return 0 ;}
```

Iteration statement/Loop (2)

Example (Two counters for loop)

```
#include <stdio.h>
int i,j;
int main() {
for(i=0,j=10; i <=5,j>=5; i++,j--){
        printf("i=%d    ", i);
        printf("j=%d \n", j);
                              }
return 0; }
```

0	10
1	9
2	8
3	7
4	6
5	5

Iteration statement/Loop (3)

Example (Nasted for loops)

```
#include <stdio.h>
int i,j;
int main() {
for(i=0; i <=5; i++){
        printf("%d       ", i);
        for(j=0; j<=5; j++){
             printf("%d ", j); }
              printf("\n");
                }
return 0; }
```

```
0       0 1 2 3 4 5
1       0 1 2 3 4 5
2       0 1 2 3 4 5
3       0 1 2 3 4 5
4       0 1 2 3 4 5
5       0 1 2 3 4 5
```

Iteration statement/Loop (4)

Example (while() and do {} while() loop)

```
#include   <stdio.h>
 int  n,F,i;
int main (){
do  {
      printf("donner un nombre:");
      scanf("%d",&n) ;
            } while (n<0) ;

      i=1 ;
      F=1 ;
      while (i<=n)
      {
      F=F*i ;
      i++ ;
      }
      printf("le factoriel de %d est %d",n,F) ;
      return 0 ;}
```

Break and Continue statement

- **The break** statement ends the loop immediately when it is encountered.

Note: The break statement is almost always used with if...else statement inside the loop.

- **The continue** statement skips the current iteration of the loop and continues with the next iteration.

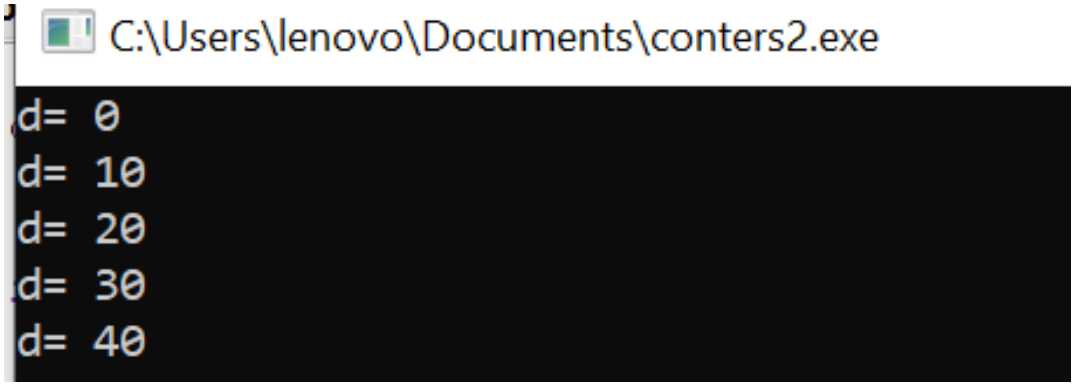
Note: The continue statement is almost always used with the if...else statement.

Break and Continue statement

Example Break;

```
#include  <stdio.h>

int  d ;
    int main (){
        for (d=0; d<=100; d+=10) {
            if(d == 50) {
                break;
                }
    printf("d= %d \n", d);
        }
  return 0 ;}
```



Break and Continue statement (2)

Example continue;

```
#include   <stdio.h>

   int  d ;
        int main () {
        for (d=100; d>=0; d-=10){
                if(d == 50) {
                    continue; }
        printf("d= %d \n", d);
                    }
  return 0 ;}
```