

LANGUAGE C

Quelle est la différence entre un programme et un algorithme ?

Définition d'algorithme. La **différence entre un algorithme et un programme** est souvent une question de niveau de détail. Un **algorithme** est souvent exprimé avec une notation indépendante de tout langage de **programmation** alors qu'un **programme** est écrit dans un langage de **programmation** particulier.

exercice1

Écrire un programme qui lit un caractère au clavier et affiche le caractère ainsi que son code numérique en employant getchar et printf,

solution 1

```
#include <stdio.h>
main()
{ int C ;
printf("introduire un caractère suivi de 'Enter'\n"); C = getchar(); printf("Le caractère %c a le code ASCII %d\n", C, C); retu
```

```
#include <stdio.h>
main()
{
    int C ;
    printf("introduire un caractère suivi de 'Enter'\n");
    C = getchar();
    printf("Le caractère %c a le code ASCII %d\n", C, C);
    return 0;
}
```

ensuite :

```
root@benam:~# gcc test.c -o test
test.c:2:1: warning: return type defaults to 'int' [-Wimplicit-int]
main()
^~~~~
root@benam:~#
```

j'exécute le programme :

```
root@benam:~# ./test
introduire un caractère suivi de 'Enter'
```

et je donne un caractere exemple 3. on me dit le caractere 3 a le code BENAM 51

```
root@benam:~# ./test
introduire un caractère suivi de 'Enter'
3
Le caractère 3 a le code BENAM 51
root@benam:~#
```

EXERCICE2 :

Ecrire un programme qui calcule et affiche la distance DIST (type double) entre deux points A et B du plan dont les coordonnées (XA, YA) et (XB, YB) sont entrées au clavier comme entiers.

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

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

```
main() { int XA, YA, XB, YB; double DIST; /* Attention: La chaîne de format que nous utilisons */
/* s'attend à ce que les données soient séparées par */ /* une virgule lors de l'entrée. */
printf("Entrez les coordonnées du point A : XA,YA "); scanf("%d,%d", &XA, &YA); printf("Entrez
les coordonnées du point B : XB,YB "); scanf("%d,%d", &XB, &YB); DIST=sqrt(pow(XA-
XB,2)+pow(YA-YB,2)); printf("La distance entre A(%d,%d) et B(%d, %d) est %.2f\n",XA, YA,
XB, YB, DIST); return 0; }
```

solution 2 :

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

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

```
main()
```

```
{
```

```
int XA, YA, XB, YB;
```

```
double DIST;
```

```
/* Attention: La chaîne de format que nous utilisons */
```

```
/* s'attend à ce que les données soient séparées par */
```

```
/* une virgule lors de l'entrée. */
```

```
printf("Entrez les coordonnées du point A:XA,YA \n");
```

```
scanf("%d,%d", &XA, &YA);
```

```
printf("Entrez les coordonnées du point B:XB,YB \n");
```

```
scanf("%d,%d", &XB, &YB);
```

```
DIST=sqrt(pow(XA-XB,2)+pow(YA-YB,2));
```

```
printf("La distance entre A(%d, %d) et B(%d, %d) est %.2f ",XA, YA, XB, YB, DIST);
```

```
return 0;
}
```

EXERCICE 3 :

Calculez la somme des N premiers termes de la série harmonique : $1 + 1/2 + 1/3 + \dots + 1/N$

Solution 3

```
include <stdio.h>
main()
{
    int N; /* nombre de termes à calculer */
    int I; /* compteur pour la boucle */
    float SOM; /* Type float à cause de la précision du résultat. */

    do
    {
        printf ("Nombre de termes: ");
        scanf ("%d", &N);
    }while (N<1);
    for (SOM=0.0, I=1 ; I<=N ; I++)
        SOM += (float)1/I;
    printf("La somme des %d premiers termes est %f \n", N, SOM);
    return 0;
}
```

```
#include <stdio.h>
main()
{
    int N; /* nombre de termes à calculer */
    int I; /* compteur pour la boucle */
    float SOM; /* Type float à cause de la précision du résultat. */

    do
    {
        printf ("Nombre de termes: ");
        scanf ("%d", &N);
    }while (N<1);
    for (SOM=0.0, I=1 ; I<=N ; I++)
        SOM += (float)1/I;
    printf("La somme des %d premiers termes est %f \n", N, SOM);
    return 0;
}
```

ensuite :

```

root@benam:~# gcc moon.c -o moon
moon.c:2:1: warning: return type defaults to 'int' [-Wimplicit-int]
main()
~~~~~
root@benam:~# ./moon
Nombre de termes: 4

```

```

root@benam:~# ./moon
Nombre de termes: 4
La somme des 4 premiers termes est 2.083333
root@benam:~#

```

EXERCICE 4 :

Affichez un triangle isocèle formé d'étoiles de N lignes (N est fourni au clavier).

Solution 4 :

```

#include <stdio.h>
main()
{
    int LIG; /* nombre de lignes */
    int L; /* compteur des lignes */
    int ESP; /* nombre d'espaces */
    int I; /* compteur des caractères */
    do
    {
        printf("Nombres de lignes : ");
        scanf("%d", &LIG);
    }while (LIG<1 || LIG>20);

    for (L=0 ; L<LIG ; L++)
    {
        ESP = LIG-L-1;
        for (I=0 ; I<ESP ; I++)
            putchar(' ');
        for (I=0 ; I<2*L+1 ; I++)
            putchar('*');
        putchar('\n');
    }
}

```

```

}
return 0;
}

```

```

#include <stdio.h>
main()
{
    int LIG; /* nombre de lignes */
    int L; /* compteur des lignes */
    int ESP; /* nombre d'espaces */
    int I; /* compteur des caractères */
    do
    {
        printf("Nombres de lignes : ");
        scanf("%d", &LIG);
    }while (LIG<1 || LIG>20);

    for (L=0 ; L<LIG ; L++)
    {
        ESP = LIG-L-1;
        for (I=0 ; I<ESP ; I++)
            putchar(' ');
        for (I=0 ; I<2*L+1 ; I++)
            putchar('*');
        putchar('\n');
    }
    return 0;
}

```

je valide ,on me demande le nombre de ligne pour le triangle et je donne 10 (signifie 10 lignes)

```

root@benam:~# gcc moon.c -o programme1
moon.c:2:1: warning: return type defaults to 'int' [-Wimplicit-int]
main()
^~~~~
root@benam:~# ./programme1
Nombres de lignes : 10
      *
     ***
    *****
   *********
  ***********
 *****
*****
*****
*****
*****
root@benam:~# █

```

EXERCICE 5 :

a) Calculez la racine carrée X d'un nombre réel positif A par approximations successives en utilisant la relation de récurrence suivante: $X_{J+1} = (X_J + A/X_J) / 2$ $X_1 = A$ La précision du calcul J est à entrer par l'utilisateur.

b) Assurez-vous lors de l'introduction des données que la valeur pour A est un réel positif et que J est un entier naturel positif, plus petit que 50.

c) Affichez lors du calcul toutes les approximations calculées :

La 1ère approximation de la racine carrée de ... est ...

La 2e approximation de la racine carrée de ... est ...

La 3e approximation de la racine carrée de ... est ...

solution 5 :

```
#include <stdio.h>
main()
{
    double A; /* donnée */
    double X; /* approximation de la racine carrée de A */
    int N; /* degré/précision de l'approximation */
    int J; /* degré de l'approximation courante */
    do
    {
        printf("Entrer le réel positif A : ");
        scanf("%lf", &A);
    }while(A<0);
    do
    {
        printf("Entrer le degré de l'approximation : ");
        scanf("%d", &N);
    }
    while(N<=0 || N>=50);

    for(X=A, J=1 ; J<=N ; J++)
    {
        X = (X + A/X) / 2;
        printf("La %2d%s approximation de la racine carrée"
            " de %.2f est %.2f\n", J, (J==1)?"ère":"e", A, X);
    }
    return 0;
}
```

```

#include <stdio.h>
main()
{
    double A; /* donnée */
    double X; /* approximation de la racine carrée de A */
    int N; /* degré/précision de l'approximation */
    int J; /* degré de l'approximation courante */
    do
    {
        printf("Entrer le réel positif A : ");
        scanf("%lf", &A);
    }while(A<0);
    do
    {
        printf("Entrer le degré de l'approximation : ");
        scanf("%d", &N);
    }
    while(N<=0 || N>=50);

    for(X=A, J=1 ; J<=N ; J++)
    {
        X = (X + A/X) / 2;
        printf("La %2d%s approximation de la racine carrée"
               " de %.2f est %.2f\n", J, (J==1)?"ère":"e", A, X);
    }
    return 0;
}

```

```

root@benam:~# gcc algo.c -o programme2
algo.c:2:1: warning: return type defaults to 'int' [-Wimplicit-int]
main()
^~~~~~
root@benam:~# █

```

```

root@benam:~# ./programme2
Entrer le réel positif A : 16.5
Entrer le degré de l'approximation : 3
La 1ère approximation de la racine carrée de 16.50 est 8.75
La 2e approximation de la racine carrée de 16.50 est 5.32
La 3e approximation de la racine carrée de 16.50 est 4.21
root@benam:~#

```

```

root@benam:~# ./programme2
Entrer le réel positif A : 25
Entrer le degré de l'approximation : 3
La 1ère approximation de la racine carrée de 25.00 est 13.00
La 2e approximation de la racine carrée de 25.00 est 7.46
La 3e approximation de la racine carrée de 25.00 est 5.41
root@benam:~#

```

exercice 6 :

Affiche la table des produits pour N variant de 1 à 10 :

X*Y I	0	1	2	3	4	5	6	7	8	9	10
0 I	0	0	0	0	0	0	0	0	0	0	0
1 I	0	1	2	3	4	5	6	7	8	9	10
2 I	0	2	4	6	8	10	12	14	16	18	20
3 I	0	3	6	9	12	15	18	21	24	27	30
4 I	0	4	8	12	16	20	24	28	32	36	40
5 I	0	5	10	15	20	25	30	35	40	45	50
6 I	0	6	12	18	24	30	36	42	48	54	60
7 I	0	7	14	21	28	35	42	49	56	63	70
8 I	0	8	16	24	32	40	48	56	64	72	80
9 I	0	9	18	27	36	45	54	63	72	81	90
10 I	0	10	20	30	40	50	60	70	80	90	100

solution6 :

```

#include <stdio.h>
main()
{
    const int MAX = 10; /* nombre de lignes et de colonnes */
    int I; /* compteur des lignes */

```



```

int J; /* compteur des colonnes */
/* Affichage de l'en-tête */
printf(" X*Y I");
for (J=0 ; J<=MAX ; J++)
printf("%4d", J);
printf("\n");
printf("-----");
for (J=0 ; J<=MAX ; J++)
printf("----");
printf("\n");

/* Affichage du tableau */
for (I=0 ; I<=MAX ; I++)
{
printf("%3d I", I);
for (J=0 ; J<=MAX ; J++)
printf("%4d", I*J);
printf("\n");
}
return 0;
}

```

```

#include <stdio.h>
main()
{
    const int MAX = 10; /* nombre de lignes et de colonnes */
    int I; /* compteur des lignes */
    int J; /* compteur des colonnes */
    /* Affichage de l'en-tête */
    printf(" X*Y I");
    for (J=0 ; J<=MAX ; J++)
        printf("%4d", J);
    printf("\n");
    printf("-----");
    for (J=0 ; J<=MAX ; J++)
        printf("----");
    printf("\n");

    /* Affichage du tableau */
    for (I=0 ; I<=MAX ; I++)
    {
        printf("%3d I", I);
        for (J=0 ; J<=MAX ; J++)
            printf("%4d", I*J);
        printf("\n");
    }
    return 0;
}

```

```

root@benam:~# nano algo.c
root@benam:~# gcc algo.c -o programme2
algo.c:2:1: warning: return type defaults to 'int' [-Wimplicit-int]
main()
^~~~
root@benam:~# ./programme2
X*Y I   0   1   2   3   4   5   6   7   8   9  10
-----
0 I   0   0   0   0   0   0   0   0   0   0   0
1 I   0   1   2   3   4   5   6   7   8   9  10
2 I   0   2   4   6   8  10  12  14  16  18  20
3 I   0   3   6   9  12  15  18  21  24  27  30
4 I   0   4   8  12  16  20  24  28  32  36  40
5 I   0   5  10  15  20  25  30  35  40  45  50
6 I   0   6  12  18  24  30  36  42  48  54  60
7 I   0   7  14  21  28  35  42  49  56  63  70
8 I   0   8  16  24  32  40  48  56  64  72  80
9 I   0   9  18  27  36  45  54  63  72  81  90
10 I  0  10  20  30  40  50  60  70  80  90 100

```

exercice 7 :

Ecrire un programme qui saisit la dimension N d'un tableau de int (le tableau est initialement défini avec une taille maximum MAX que N ne doit pas excéder) remplit le tableau par des valeurs entrées au clavier et l'affiche. Le programme doit ensuite effacer toutes les occurrences de la valeur 0 dans le tableau, tasser les éléments restants et afficher le tableau ainsi modifier. Pour cela écrire les fonctions suivantes : void SaisirTableau (int *Tab, int N) ; void AfficherTableau(int *Tab, int N) ; int TasserTableau(int *Tab , int N) ;

solution7 :

```

#include <stdio.h>
#define MAX 50
void SaisirTableau(int *, int ) ;
void AfficherTableau(int *, int) ;
int TasserTableau(int *, int) ;
main()
{
    /* Déclarations */
    int T[MAX]; /* tableau donné */
    int N,M; /* dimension */
    /* Saisie de la dimension */
    do
    {
        printf("Dimension du tableau (max.%d) : ",MAX);
        scanf("%d", &N );
    }while(N>MAX) ;
    /* Saisie des données */
    SaisirTableau(T,N) ;

    /* Affichage du tableau */
    AfficherTableau(T,N) ;
    /*Tasser les elements du tableau */
    M = TasserTableau(T,N) ;
    /* Edition des résultats */

```

```

AfficherTableau(T ,M) ;
}
void SaisirTableau(int *Tab, int N)
{
    int i ;
    for (i=0; i<N; i++)
    {
        printf("Elément %d : ", i);
        scanf("%d", &Tab[i]) ;
    }
}
void AfficherTableau(int *Tab, int N)
{
    int i ;
    printf("Tableau donné : \n");
    for (i=0; i<N; i++)
        printf("%d ", Tab[i]);
    printf("\n");
}
int TasserTableau(int * Tab, int N)
{
    int i,j ;
    /* Effacer les zéros et compresser : */
    /* Copier tous les éléments de i vers j et */
    /* augmenter j pour les éléments non nuls. */
    for (i=0, j=0 ; i<N ; i++)
    {
        Tab[j] = Tab[i] ;
        if (Tab[i])
            j++ ;
    }
    /* La nouvelle dimension du tableau est retournée */
    return j ;
}

```

```

root@benam:~# gcc algo.c -o code
algo.c:6:1: warning: return type defaults to 'int' [-Wimplicit-int]
main()
^
root@benam:~# ./code
Dimension du tableau (max.50) : 10
Elément 0 : 4
Elément 1 : 9
Elément 2 : 8
Elément 3 : 7
Elément 4 : 6
Elément 5 : 5
Elément 6 : 4
Elément 7 : 3
Elément 8 : 2
Elément 9 : 1
Tableau donné :
4 9 8 7 6 5 4 3 2 1
Tableau donné :
4 9 8 7 6 5 4 3 2 1
root@benam:~# █

```

exercice 8 :

Ecrire un programme qui lit deux chaînes de caractères, et qui indique leur précedence lexicographique dans le code de caractères de la machine (ici: code ASCII). On écrira pour cela la fonction `precedence` qui récupère les deux chaînes en paramètre et qui retourne 1 si la première chaîne précède la deuxième, 2 si la deuxième précède la première, 0 si elle sont égale. Prototype :

```
int precedence(char *,char *)
```

solution 8 :

```

#include <stdio.h>
int precedence(char *,char *) ;
main()
{
    /* Déclarations */
    char CH1[50], CH2[50]; /* chaînes à comparer */
    int r ;
    /* Saisie des données */
    printf("Entrez la première chaîne à comparer : ");
    gets(CH1);
    printf("Entrez la deuxième chaîne à comparer : ");
    gets(CH2);
    r = precedence (CH1,CH2) ;
    if(r==0)
        printf("\"%s\" est égal à \"%s\"\n", CH1, CH2);
    else if (r == 1)
        printf("\"%s\" précède \"%s\"\n", CH1, CH2);
    else
        printf("\"%s\" précède \"%s\"\n", CH2, CH1);
}
int precedence (char *CH1,char *CH2)

```

```

{
    int I; /* indice courant */
    int r ;
    for (I=0; (CH1[I]==CH2[I]) && CH1[I] && CH2[I]; I++) ;

    if (CH1[I]==CH2[I])
        r = 0 ;
    else if (CH1[I]<CH2[I])
        r = 1 ;
    else
        r = 2 ;
    return r;
}

```

```

root@benam:~# gcc langage.c -o code1
langage.c:3:1: warning: return type defaults to 'int' [-Wimplicit-int]
main()
^~~~~
langage.c: In function 'main':
langage.c:10:2: warning: implicit declaration of function 'gets'; did you mean 'fgets'? [-Wimplicit-func
tion-declaration]
    gets(CH1);
    ^~~~~
    fgets
/tmp/cc9WozeX.o : Dans la fonction « main » :
langage.c:(.text+0x38): avertissement : the `gets' function is dangerous and should not be used.
root@benam:~# ./code1
Entrez la première chaîne à comparer : 50
Entrez la deuxième chaîne à comparer : 25
"25" précède "50"
root@benam:~# █

```

```

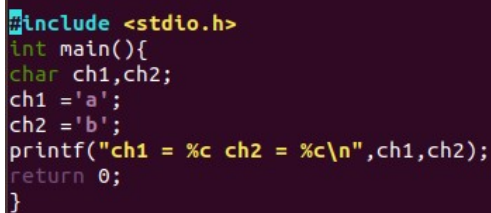
root@benam:~# ./code1
Entrez la première chaîne à comparer : 50
Entrez la deuxième chaîne à comparer : 25
"25" précède "50"
root@benam:~# ./code1
Entrez la première chaîne à comparer : m
Entrez la deuxième chaîne à comparer : w
"m" précède "w"
root@benam:~# █

```

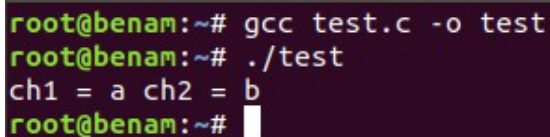
EXO1 1 :

nous allons presenter quelques variables

```
#include <stdio.h>
int main(){
char ch1,ch2;
ch1 ='a';
ch2 ='b';
printf("ch1 = %c ch2 = %c\n",ch1,ch2);
return 0;
}
```



```
#include <stdio.h>
int main(){
char ch1,ch2;
ch1 ='a';
ch2 ='b';
printf("ch1 = %c ch2 = %c\n",ch1,ch2);
return 0;
}
```



```
root@benam:~# gcc test.c -o test
root@benam:~# ./test
ch1 = a ch2 = b
root@benam:~#
```

EXO2 :

LES ENTIERS DE TYPE INT :

j'ai donné a nombre1 =200 et nombre2=40

```
#include <stdio.h>
int main(){
int nbre1,nbre2,som,mul,div;
nbre1 =200;
nbre2 =40;
som = nbre1+nbre2;
mul = nbre1*nbre2;
div = nbre1/nbre2;
printf("la somme est %d\n",som);
printf("la multiplication est %d et la division est %d\n",mul,div);
return 0;
}
```

voici le programme a effectué les résultats comme :

la somme est égale a 2040

la multiplication est égale a 8000

la division est égale a 5

```
root@benam:~# gcc vision.c -o vision
root@benam:~# ./vision
la somme est =240
la multiplication est 8000 et la division est 5
```

EXO3 :

passons aux reels :

on va faire un petit programme de calculatrice

```
#include <stdio.h>
int main(){
float reel1,reel2;
printf("entrer deux nombres reels\n");
scanf("%f",&reel1);
scanf("%f",&reel2);
float som = reel1+reel2;
printf("la somme est %f\n",som);
return 0;
}
```

j'ai entré un réel 15.2 et ensuite je donne le deuxième réel qui est 10.5 et le programme va faire la somme qui est 25.700001

```
root@benam:~# gcc algo.c -o algo
root@benam:~# ./algo
entrer deux nombres reels
15.2
10.5
la somme est 25.700001
root@benam:~#
```

etape2 :

je vais modifier le paramètre au lieu de float(réel) mais plus tôt int(entier) :

```
#include <stdio.h>
int main(){
int entier1,entier2;
printf("entrer deux nombres entiers\n");
scanf("%d",&entier1);
scanf("%d",&entier2);
int som = entier1+entier2;
printf("la somme est %d\n",som);
```

```
return 0;
```

```
}
```

A terminal window with a dark purple background. The prompt is 'root@benam:~#'. The user enters './algo', and the program outputs 'entrer deux nombres entiers'. The user enters '2500', and the program outputs '5000'. The user enters 'la somme est 7500', and the program outputs 'root@benam:~#'.

```
root@benam:~# ./algo
entrer deux nombres entiers
2500
5000
la somme est 7500
root@benam:~#
```

« BERENGER_BENAM »