Programmation en langage C

Partie 1 - les notions de base

1. Notions de base de programmation
2. Notion de variable et les types de base
3. Les fonctions d'entrées-sorties standard
4. Syntaxe du langage C – les opérateurs
5. Les structures de contrôle
6. Les structures répétitives – les boucles

Un programme est une séquence d'instructions, exécuté une après l'autre en général de manière séquentielle.

Pour le langage C, les instructions sont stockées dans un fichier texte d'extension **".c"** par suite ce fichier sera traduit en langage machine par un compilateur puis exécuté.

Certain instructions et fonctionnalités, par exemple d'affichage, de lecture, d'accès aux matérielles, d'accès au réseau, etc. sont déjà définit dans des bibliothèques, on fait seulement appel à ces fonctionnalités.

```
stdio.h contient les fonctions d'affichage et de lectures
math.h contient les fonctions mathématiques cos, sin, exp,...
```

string.h contient les fonctions de manipulation des chaines de caractères
Et bien d'autres ...

```c
#include <stdio.h>  // directives du préprocesseur
#include <stdlib.h>
#define carre(x) x*x // déclaration d'alias
#define N 150

float prix=12.5; // déclaration des variables globales
// Déclaration d'une fonction
int somme(int a,int b){ return a+b;}
// Déclaration d'une fonction principale main
int main(){
    // Les instructions
```

```
    int a=1,b=3,s; // Déclaration des variables
    locales
    s=somme(a,b); // appel du fonction somme // appel
    du fonction d'affichage
    printf printf("la somme est %d",s);
    return 0;
}    // Fin du programme
```

Les commentaires sont des parties du programme qui ne seront ni compilées ni exécutées, ils servent a ajouter une documentation ou explication.

```
/* Commentaire a
   plusieurs lignes
*/
// Commentaire à la fin de ligne
```

Pour définir les noms des variables, les noms des fonctions, les noms des structures.

```
float prix=12.5;
```

```
Prix prix _prix prix01 PRIX01 _0prix_      valide
0prix &prix pr-ix +prix +prix-784          non valide
```

Les noms suivants sont réservés au langage :

```
auto break case char const continue default do double else
enum extern float for goto if int long register return
```

```
short  signed  sizeof  static  struct  switch  typedef  union
unsigned void volatile while
```

Une variable est une case mémoire pouvant contenir des données (valeurs) d'un type défini, ses valeurs sont changeables dans le temps.

```
type nomDeVariable1,nomVariable2;
type nomDeVariable = valeurInitialisation;
```

Exemples :

```
int nombre, age=18, x=-4;
float poids=12.4, pi=3.14159;
```

```c
char c='F'; // caractère du table code ASCII
int a=124; // entier
long l=1554220; // entier long
float f=3.14159; // réel avec 6 chiffres décimaux
double d=3.1415926535; // réel avec double précision
15 chiffres décimaux
long double ld=3.141592653589; // 19 chiffre décimaux
```

Pour le type boolien : une variable **`int`** (entier) peut jouer le rôle d'une variable boolien s'elle est égale à `0` => `faux` et s'elle est différente de `0` => `vrai`

Il existe d'autres types de base comme **`short int, unsigned int, long long ...`**

1 - Les formats de types

Prennent l'exemple suivant :

```
#include <stdio.h>
int main()
{ int a=12;
printf("la valeur de a est %d",a);
    //affichage de la valeur de a
    return 0;
}
```

Pour afficher la valeur de `a` on a utilisé `%d` qui sera remplacé par la valeur du variable a dans l'appel du fonction d'affichage, `%d` appelé format de type entier.

1 - Les formats de types

Pour afficher d'autres types, on à la table suivante :

`%c`	**`char`**	`'c' 'E' '1' '\t'`
`%d`	**`int`**	`1 0 -2 31`
`%ld`	**`long int`**	`142000 78000 900000`
`%f`	**`float`**	`0.4 3.5 -.78 0.01`
`%f`	**`double`** `5.0 3.5`	
`%lf`	**`long double`**	`2.1e4 3.14159263589`
`%s`	**`char*`**	**`"chaine" "Karim Mahmoud"`**

2 – Affichage et saisie printf :

permet l'affichage sur l'écran

```
printf("valeur est %d",15);
printf("%6.5f",1256.46587689)
; //  1256.46587 forme
décimale
printf("%e",3.1415926);
// 1.245125e+003 forme exponentiel
```

scanf : permet de lire du clavier et stocker dans un variable de même type que le format.

Les fonctions d’entrées-sorties standard

```
scanf("%f",&f);
scanf("%c",&c);
```

1 – Les opérateurs arithmétiques

a+b addition

a-b soustraction

a*b multiplication

a/b division

a%b modulo (reste de la division euclidienne de a et b)

a++ ou ++a incrémentation de a avec 1 (a=a+1)

a-- ou --a décrémentation de a avec 1 (a=a-1)

2 – Affectation

```
x=y    x reçoit la valeur de y
x+=y   (x=x+y) x reçoit la valeur de x+y
x-=y    x*=y    x/=y    x%=y    x*=y
```

3 – Relation

```
x==y égalité donne 1 si x est égale à y; 0 sinon
x!=y inégalité donne 1 si x est diffèrent a y; 0 sinon
x <y inférieur   donne 1 si x < y; 0 sinon
x >y supérieur   donne 1 si x > y; 0 sinon
x<=y inférieur ou égale    donne 1 si x <= y; 0 sinon
x>=y supérieur ou égale   donne 1 si x >= y; 0 sinon
```

–

4 – Les opérateurs logiques

```
&& et logique
|| ou logique
!  non logique
```

Exemples : a doit être diffèrent de zéro et inférieur ou égale à b ou simplement b supérieur à 20

```
(a != 0 && a <= b ) || b > 20
```

1 if else (si sinon)

```
if(condition) {
      Bloc d'instructions 1;
} else {
      Bloc d'instructions 2;
}
```

Exemple : saisie d'une valeur, et l'afficher s'il est nulle ou non.

```
int a;
printf("donner a");
scanf("%d",&a);
if(a==0)
{ printf("a est nul");
}
else
{ printf("a est non nul");
}
```

–

2 else if (sinon si)

```
if(condition 1) {
      Bloc d'instructions 1;
} else if(condition 2) {
      Bloc d'instructions 2;
} else {
      Bloc d'instructions 3;
}
```

Exemple : saisie d'une valeur, et l'afficher s'il est nulle, négative ou positive.

```
int a;
printf("donner a");
scanf("%d",&a); if(a==0){
printf("a est nul");
```

–

```
} else if(a>0){ printf("a est
    positif ");
} else if(a<0){ printf("a
    est negatif");
}
```

3 switch (choix)

```
switch(valeur)
    { case valeur1: bloc d'instruction 1;
        break ; case valeur2: bloc
        d'instruction 2; break ;
        ... ... default: bloc d'instruction des
    autres cas; }
```

L'instruction **break** permet de passer les autres cas si une est exécutée.

3 switch (choix)

Exemples : demande un numéro et affiche le jour de la semaine correspondant.

—

```c
int a;
printf("donner numero de jour ");
scanf("%d",&a);
switch(a){
case 1: printf("lundi"); break;
case 2: printf("mardi"); break;
case 3: printf("mercredi"); break ;
case 4: printf("jeudi"); break;
case 5: printf("vendredi");break;
case 6: printf("samedi"); break ;
case 7: printf("dimanche"); break;
    default:printf("n'est pas un jour");
}
```

–

1 while (tant que)

Tant que la condition est satisfaite (vrai) le bloc d'instruction sera exécuté

```
while(condition){
    Bloc d'instructions;
}
```

Exemple : affichage des valeurs de 0 à 5

```
// Décroissant
// Donne 5 4 3 2 1 0
i=5;
while(i>=0){
    printf("i=%d\n",i);
    i--; // i=i-1;
}
```

```
// Croissant
// Donne 0 1 2 3 4 5
i=0;
while(i<=5){
    printf("i=%d\n",i);
    i++; // i=i+1;
}
```

2 do while (répéter)

Répéter l'exécution du bloc d'instructions tant que la condition est vraie.

```
do {
    Bloc d'instructions;
} while(condition);
```

Exemple : affichage des valeurs de 0 à 5

```
// Donne 0 1 2 3 4 5
i=0; do {
printf("i=%d\n",i);
i++;
} while(i<=5);
```

–

3 for (pour)

Permet de répéter un bloc d'instructions pour un nombre d'itérations connu.

```
for(initialisation; condition; incrémentation){
bloc d'instructions; }
```

Exemple : affichage des valeurs de 0 à 5

```
// Donne 0 1 2 3 4 5
for(i=0;i<=5;i++){
printf("%d\t",i);
}
```

4 Exemples

Exemple 1 : afficher 5 étoiles (*) successivement

```
// Résultat : //
*****
for(i=1;i<=5;i++){
printf("*");
}
```

Exemple 2 : afficher 5 lignes et dans chaque ligne 5 étoiles (*) (après chaque ligne un retour à la ligne `"\n"`)

–

```
for(i=1;i<=5;i++){
    for(j=1;j<=5;j++){
    printf("*");
    } printf("\n");
}
```

```
// Résultat :
// *****
// *****
// *****
// *****
// *****
```

4 Exemples

Exemple 3 : afficher un triangle d'étoiles

```
for(i=1;i<=5;i++){   // *
for(j=1;j<=i;j++){ // **
printf("*");     // ***
   }
   printf("\n");
}
```

```
// Résultat :
// ****
// *****
```

Exemple 4 : afficher un triangle d'étoiles centré

```
for(i=1;i<=5;i++){                          // Résultat :
   for(j=1;j<=5-i;j++){ //      *
   printf(" "); //     ***
   }                                        //    *****
   for(j=1;j<=i;j++){                       //   *******
       printf("*");                         // *********
   } for(j=1;j<=i-1;j++){
   printf("*");
   }
   printf("\n");
}
```


statement
standard
void
array
contains
constant
variables
function
POINTER
size
argument
zero
define
int
read
char
main
version
CHARACTER
USED
INITIALIZED
PROGRAM
include
TYPE
CALLED
return
declaration
expression
external
RESULT
NULL
object
operator
double
error
input
structure
STRUCT
time
write
string
specifier
library
SPACE
left

Programmation en langage C

Partie 2 - les tableaux – les fonctions – les chaines de caractères – les pointeurs

Y.KERROUM