Atelier de programmation
Langage C

Auditoire : LGLSI 1

LIRS 1

Mahmoud Ltaief

Chapitre I

Eléments de base du langage C

1. Généralités

- Le langage C a été créé dans les années 1970 par Dennis Ritchie.
- Un programme en langage C est une suite d'instructions.
- Un programme en C est un ensemble de fonctions, parmi ces fonctions, il existe une fonction de démarrage dont le nom est "main".

2. Caractéristique du C

- Structuré
- Modulaire: peut être découpé en modules qui peuvent être compilés séparément
- Universel: n'est pas orienté vers un domaine d'application particulier
- Typé: tout objet C doit être déclaré avant d'être utilisé
- Portable: sur n'importe quel système en possession d'un compilateur C

3. Les avantages

Le grand succès du langage C s'explique par les avantages suivants ; C est un langage :

- Universel : C n'est pas orienté vers un domaine d'applications spéciales,
- Compact : C est basé sur un noyau de fonctions et d'opérateurs limités, qui permet la formulation d'expressions simples, mais efficaces.
- Moderne : C est un langage structuré, déclaratif et récursif ; il offre des structures de contrôle et de déclaration comparables à celles des autres grands langages de ce temps (FORTRAN, ALGOL68, PASCAL).
- Rapide : Comme C permet d'utiliser des expressions et des opérateurs qui sont très proches du langage machine, il est possible de développer des programmes efficients et rapides.
- Extensible : C ne se compose pas seulement des fonctions standard ; le langage est animé par des bibliothèques de fonctions privées ou livrées par de nombreuses maisons de développement.

4. Structure d'un programme en C

```
#include<stdio.h>
void main()
{
<Déclarations>
<Instructions>
}
```

Appel des fonctions de base du C existant dans le fichier stdio.h

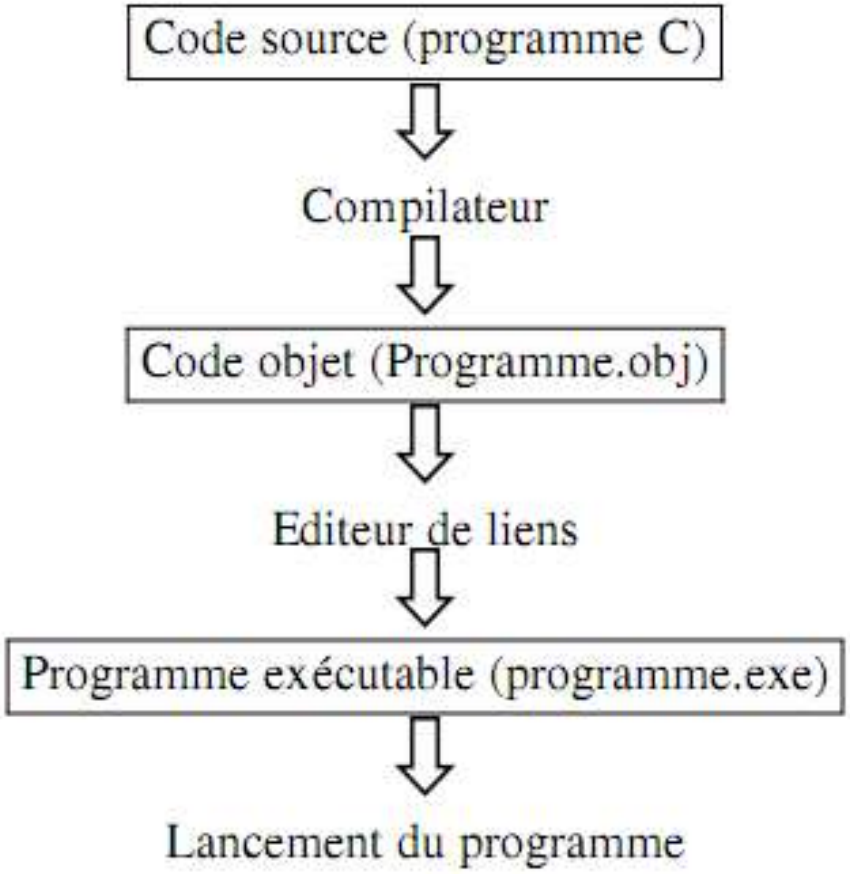
En C le programme principal s'appelle toujours *main*

void main(void): La fonction main ne prend aucun paramètre et ne retourne pas de valeur.

int main(void): La fonction main retourne une valeur entière à l'aide de l'instruction return (0 si pas d'erreur).

int main(int argc, char *argv[ ]): On obtient alors des programmes auxquels on peut adresser des arguments au moment où on lance le programme.

5. Les diverses phases d'un programme C



6. Fichiers "include"

- Se trouvent dans un répertoire nommé "include",
- Ils possèdent l'extension h,
- Ils renferment les déclarations des fonctions du système.

Exemple :

- #include<stdio .h> (Fichiers d'entrée, sortie).
- #include<math.h> (Fichiers Mathématiques).
- #include<graphics.h> (Fonctions Graphiques).

Exemple de programme

```
#include<stdio.h>
void main()
{
printf("Bonjour");
}
```

7. Les commentaires

Un commentaire est une suite de caractères placés entre des délimiteurs.

Exemple

```
/*Les commentaires documentent les programmes*/
```

Ajoutons un commentaire au programme précédant :

```
#include<stdio.h>
void main()
{
printf("Bonjour"); /*Affichage d'un message*/
}
```

8. Les types de bases

Les différents types de bases sont :

- Les nombres entiers int
- Les nombres flottants float ou double
- Les caractères char

8.1 Le type entier (int)

Type	Désignation	Espace	Plage de valeurs
entier court	short int (short)	2 octets	-32768 à 32767
entier court non signé	unsigned short	2 octets	0 à 65535
Entier (type standard)	int	2 octets	-32768 à 32767
entier non signé	unsigned int	2 octets	0 à 65535
entier long	long int (long)	4 octets	-2 147 483 648 à 2 147 483 647
entier long non signé	unsigned long	4 octets	0 à 4 294 967 295

8.1 Le type entier (int)

Remarques :

1. Les mots clés short et long permettent d'influer sur la taille mémoire des entiers et des réels.

2. Les deux mots signed et unsigned peuvent s'appliquer aux types caractère et entier pour indiquer si le bit de poids fort doit être considéré ou non comme bit de signe.

3. Par défaut, les types entiers short int et long sont munis d'un signe. De ce fait, cela revient au même d'écrire : int et signed int , short et signed short , long et signed long.

4. Le type short est en général codé dans 2 octets. Comme une variable int occupe aussi 2 octets, le type short devient seulement nécessaire, si on veut utiliser le même programme sur d'autres machines sur lesquelles le type standard des entiers (int) n'est pas forcément 2 octets.

8.2 Le type flottant (float)

Un nombre réel en C est dit nombre à virgule flottante. C'est un nombre dans lequel la position de la virgule en tant que séparateur entre partie entière et partie décimale n'est pas fixe.

Exemple :

Nombre = 14.8

Représentation 1 : $1.48 * 10^{1}$

Représentation 2 : $0.148 * 10^{2}$

Représentation 3 : $148.0 * 10^{-1}$

En général, un réel est représenté sous la forme suivante :

<signe> <mantisse> *10 <exposant>

8.2 Le type flottant (float)

avec :

<signe> : +|−

<mantisse> : décimal positif

<exposant> : entier relatif.

Ils existent trois types float :

Exemple: 1.23*10E2

-2.5*10E3

Type	**Désignation**	Nb chiffres mantisse	**Espace**
réel simple précision	float	6	4 octets
réel double précision	double	15	8 octets
réel avec précision étendue	long double	19	10 octets

8.3 Le type caractère (char)

Ils existent 2 types différents :

Type	Désignation	Espace	Plage de valeurs
caractère	char	1 octets	-128 – 127
caractère non signé	unsigned char	1 octets	0 – 255

- Il est utilisé pour représenter un caractère.
- Le type char (du mot anglais character, "caractère") et utilisé pour représenter un caractère, plus précisément la valeur entière d'un élément de l'ensemble des caractères représentables. Ce nombre entier est le code ASCII du caractère.

Exemple :

- Le caractère 'A' est stocké sous forme char
- L'ordinateur n'écrit pas le signe A, mais le nombre 65 (code ASCII de 'A') sur 1 octet = 01000001.

9. Les constantes

1ere méthode:

Définition d'un symbole à l'aide de la directive de compilation #define.

Syntaxe : #define identificateur texte(valeur)

Exemple: #define PI 3.14

2eme méthode:

Déclaration d'une variable, dont la valeur sera constante pour tout le programme.

Syntaxe : const type identificateur = texte(valeur);

Exemple: const float PI = 3.14;

9.1 Les caractères de contrôle

Séquence	Signification
\n	Génère une nouvelle ligne
\t	Pose une tabulation horizontale
\f	Provoque un saut de page
\a	Déclenche un signal sonore
\r	Provoque un "Retour chariot"

10. Les variables

Une variable :

- possède un nom,
- possède un type,
- possède un contenu,
- est modifiable,
- est rangée en mémoire à partir d'une certaine adresse.

Déclaration

Type NomVar1, NomVar2, ..., NomVarN;

Exemples :

- char c;
- int X,Y;
- float Z;
- double W;

Déclaration et initialisation

Type NomVar1= val 1, NomVar2= Val 2, ...,
NomVarN= ValN;

Exemples :

- char c='A';
- int X=5,Y=12;
- float Z = 4.5;
- double W=512.36;

11. Les opérateurs

11.1 Les opérateurs arithmétiques

- +, -, /,*
- % (Modulo : reste de la division entière).

Remarque :

- Si les deux arguments de / sont des entiers alors le résultat est un entier.
- Le résultat de x= 3/2 est 1, même si x est déclaré de type float.
- Le résultat de x= 3.0/2 (ou 3/2.0) est 1.5.
- Pas d'opérateur d'évaluation en puissance, (On utilise la fonction Pow(x,y) =x^y).

11.2 Les opérateurs de comparaison

- <, <=, >, >=,
- == (égal à), != (différent de).
- Le résultat d'une comparaison est un entier : 0 si le résultat est faux et 1 si le résultat est vrai.
- La comparaison devient une expression de type entier

11.3 Les opérateurs logiques

- && : Et (and),
- || : Ou (or),
- ! : Non (not).

Exemples :

- (a < b)&&(c < d) : Prend la valeur 1(vrai) si les deux expressions a < b et c < d sont toutes deux vraies et la valeur 0 dans le cas contraire.
- (a < b)||(c < d) : Prend la valeur 1 (vrai) si l'une au moins des deux conditions a < b et c < d est vraie et la valeur 0 dans le cas contraire.
- !(a < b) : Prend la valeur 1(vrai) si la condition a < b est fausse et la valeur 0 (faux) dans le cas contraire.

Remarques :

- !(a == b) est différent de !a == b
- i f (n) est équivalente à i f (n! = 0)
- i f (!n) est équivalente à i f (n == 0)

11.4 Les opérateurs d'affectation

- Les opérateurs d'affectation mettent dans leur opérande de gauche la valeur de leur opérande de droite.
- L'opérande de gauche est appelé une Lvalue :
- Une expression désignant une adresse dans la mémoire de l'ordinateur.
- Les affectations sont évaluées de la droite vers la gauche.

Affectation simple

Exemple 1 :

```
int x;
```

x =1; La variable x du membre droit de l'affectation reçoit la valeur 1.

Exemple 2

```
int x,y;
```

x=y=1;Affecte la valeur 1 aux deux variables (x=1 et y=1).

Affectation combinée

Les affectations combinées mélangent une opération d'affectation avec une opération arithmétique ou logique.

Exemple :

- X+=Y; est équivalente à X=X+Y;
- X-=Y; est équivalente à X=X- Y
- X*=Y; est équivalente à X=X*Y;
- X/=Y; est équivalente à X=X / Y
- X%=Y; est équivalente à X=X%Y;

11.5 Les opérateurs d'incrémentation et de décrémentation

- Les opérateurs d'incrémentation et de décrémentation unaires ++ et -- augmentent (incrémentent) ou diminuent (décrémentent) la valeur d'une variable de la quantité 1.
- x++; est équivalente à ++x; est équivalente à x=x+1.
- x-- ; est équivalente à –x; est équivalente à x=x-1.

Postfixe	Préfixe
Int x=1, y; y=x++; Résultat: x=2 et y=1	Int x=1, y; y=++x; Résultat: x=2 et y=2
Int x=1, y; y=x--; Résultat: x=0 et y=1	Int x=1, y; y=--x; Résultat: x=0 et y=0

11.6 Priorité des opérateurs

Priorité	Opérateur	Ordre d'évaluation
1 (la plus forte)	( )	de gauche à droite
2	!, ++, −−, +*unaire*, −*unaire*	de droite à gauche
3	*, /, %	de gauche à droite
4	+, −	de gauche à droite
5	<, <=, >, >=	de gauche à droite
6	==, !=	de gauche à droite
7	&&	de gauche à droite
8	||	de gauche à droite
9 (la plus faible)	=, + =, − =, * =, / =, % =	de droite à gauche

12. Fonctions arithmétiques standards

Les fonctions suivantes sont prédéfinies dans la bibliothèque standard <math.h>. Pour pouvoir les utiliser, le programme doit contenir la ligne : #include <math.h>

Nom	Explication
log(X)	logarithme naturel
log10(X)	logarithme à base 10
exp(X)	fonction exponentielle
pow(X,Y)	X exposant Y
sqrt(X)	racine carrée de X
fabs(X)	valeur absolue de X
floor(X)	arrondir en moins
ceil(X)	arrondir en plus
fmod(X,Y)	reste rationnel de X/Y pour X différent de 0
sin(X) cos(X) tan(X)	sinus, cosinus, tangente de X
asin(X) acos(X) atan(X)	arcsin(X), arccos(X), arctan(X)

13. Les expressions et les instructions

Les expressions

- Les expressions sont des combinaisons entre des variables et des constantes à l'aide d'opérateurs. Elles expriment un calcul (expressions arithmétiques) ou une relation (expressions logiques).
- Une expression comme i=0 ou x++ ou printf(...) devient une instruction, si elle est suivie d'un point-virgule.

Exemples:

$$i = 0;$$
$$i++;$$
$$X = pow(A, 4);$$
$$printf("Bonjour!\n");$$
$$a = (5 * x + 10 * y) * 2;$$

14. Les entées-sorties

La bibliothèque standard < stdio.h > contient un ensemble de fonctions qui assurent la communication de la machine avec le monde extérieur. Dans ce chapitre ; nous allons nous en discuter des plus importantes :

- printf() écriture formatée de données
- scanf() lecture formatée de données
- putchar() écriture d'un caractère
- getchar() lecture d'un caractère

14.1. Ecriture formatée de données

La fonction "printf" est utilisée pour transférer du texte, des valeurs de variables ou des résultats d'expressions vers l'écran.

Syntaxe :

printf("<Format>", <Expr1>, <Expr2>, ..., <ExprN>);

- " < Format > " : Format de représentation
- < Expr1 >, ... : variables et expressions dont les valeurs sont à représenter.
- La partie " < Format > " est en fait une chaîne de caractères qui peut contenir : Du texte, des séquences d'échappement et/ou des spécificateurs de format.
- Les spécificateurs de format indiquent la manière dont les valeurs des expressions <Expr1>,…,< ExprN > sont imprimées.
- La partie " < Format > " contient exactement un spécificateur de format pour chaque expression < Expr1 >,…,< ExprN >.

Exemple1 :

- printf ("%d",5) ;

Résultat : 5

- int i = 5;

printf ("%d", i);

Résultat : 5

- printf ("La somme 2+3 donne %d",2+3);

Résultat : La somme 2+3 donne 5

- int i = 5;

printf ("%d plus %d donne : %d", 1000, i , i+1000);

Résultat : 1000 plus 5 donne : 1005.

Spécificateurs de format pour "printf"

Format	Objet de données
%d ou %i	Nombre relatif.
%u	Nombre entier non signé.
%o	Nombre exprimé en octal.
%x	Nombre exprimé en hexadécimal.
%c	Caractère.
%f	Nombre à virgule flottante.
%e, %E	Nombre à virgule flottante en format exponentiel.
%s	Chaîne de caractères.

Remarque :

Par défaut, les nombres sont affichés avec le nombre de caractères nécessaires.

Format : % nb d

Si nb>nombre de caractères alors Ajout du blanc

14.2. Lecture formatée de données

La fonction "scanf" est la fonction symétrique à printf ; elle nous offre pratiquement les mêmes conversions que printf, mais en sens inverse.

Syntaxe : scanf("<Format>", &Var1, &Var2,... , &VarN);

- ” < Format > ” : Format de lecture des données.
- < &Var1 > : adresses des variables auxquelles les données seront attribuées.
- La chaîne de format détermine comment les données reçues correctement sont mémorisées successivement aux adresses indiquées par < &Var1 >,< &Var2 >,….
- L’adresse d’une variable est indiquée par le nom de la variable précédé du signe &.

Exemple 1:

```
int x;
scanf("%d",&x);
```

L'opérateur d'adressage & applique au nom de la variable, il informe donc la fonction "scanf" de l'emplacement de stockage de la valeur saisie.

Exemple 2:

Lors de l'entrée des données, une suite de signes d'espacement (espaces, tabulateurs, interligne est évaluée comme un seul espace. Pour la suite d'instructions :

```
int jour, mois, annee ;
scanf("%d%d%d",&jour,&mois,&annee);
```

Les entrées suivantes sont correctes et équivalentes :

- 22 10 2019
- 22
  10
  2019

14.3. La macro "getchar"

- Une macro est un nom qui représente une ou plusieurs instructions ou expressions.
- La macro "getchar" lit un caractère isolé depuis le clavier et le met à la disposition du programme.

Exemple :

```
# include<stdio.h>
void main()
{
char carlu;
printf ("Entrez un caractère. \n");
carlu= getchar(); /*Lecture d'un caractère via "getchar"
et affectation du caractère à la variable carlu */
printf("\n le caractère saisi est %c:", carlu);
}
```

14.4. La macro "putchar "

- La macro "putchar" affiche un caractère non formaté sur l'écran.
- La donnée à afficher est écrite sous forme de paramètre entre les parenthèses de la macro.

Exemple :

```
#include <stdio.h>
void main()
{
char c;
printf("Entrez une lettre \n");
c = getchar();
putchar (c);
}
```

15. Les conversions forcées de type (casting)

Il est possible de convertir explicitement une valeur en un type quelconque en forçant la transformation à l'aide de la syntaxe :

(<Type>) <Expression>

Exemple

Nous divisons deux variables du type entier. Pour avoir plus de précision, nous voulons avoir un résultat de type rationnel. Pour ce faire, nous convertissons l'une des deux opérandes en float. Automatiquement C convertira l'autre opérande en float et effectuera une division rationnelle :

```
int A=3;

int B=4;

float C;

C = (float)A/B;
```

La valeur de A est explicitement convertie en float. La valeur de B est automatiquement convertie en float (règle 2). Le résultat de la division (type rationnel, valeur 0.75) est affecté à C.

Résultat : C=0.75

Attention !

Les contenus de A et de B restent inchangés ; seulement les valeurs utilisées dans les calculs sont converties !

Chapitre 2

Les Structures conditionnelles

1. Introduction

En programmation, on est souvent confronté à des situations où on a besoin de choisir entre 2 ou plusieurs traitements selon la réalisation ou non d'une certaine condition ; d'où la notion de traitement conditionnel.

2. La structure if ... else

Syntaxe :

```
if(<Condition>)
<Bloc d'instructions1>;
else
<Bloc d'instructions2>;
```

1. Si la <Condition> fournit une valeur différente de 0 alors le <bloc d' instructions 1> est exécuté.
2. Si la <Condition> fournit la valeur 0 alors le <bloc d' instructions 2> est exécuté.
3. La partie <Condition> peut désigner :
   - Une variable numérique.
   - Une expression fournissant un résultat numérique.
4. La partie <bloc d' instructions > peut désigner :
   - Un bloc d'instructions compris entre accolades.
   - Une seule instruction terminée par un point-virgule.

Exemple1

```
if(a>b)
    Max=a;
else
    Max=b;
```

Exemple2

```
if(A-B)
        printf ("A est différent de B\n");
else
        printf ("A est égal à B\n");
```

Exemple3

```
if (A>B)
    {
    aide=A;
    A=C;
    C=aide;
    }
else
    {
    aide=B;
    B=C;
    C=aide;
    }
```

3. La structure if sans else

Syntaxe :

```
if(<Condition>)
   <Bloc d'instructions I>;
```

Exemples :

1.
```
if ( x>y && x>0 )
    printf ("x est plus grand que y\n ");
```
2.
```
if (nb==0)
    printf ("%d ne peut pas être un diviseur \n", nb) ;
```

4. Structure conditionnelle Imbriquée

Syntaxe :

```
if(<Cond 1>)
        <Bloc 1>;
else
        if (<Cond 2>)
            <Bloc 2>;
        else
        .
        .
                    if (<Cond N>)
                        <Bloc N>;
                    else
                        <Bloc N+1>;
```

- Les conditions <Cond 1> ... <Cond N> sont évaluées du haut vers le bas jusqu'à ce que l'une d'elles soit différente de 0. Le bloc d'instructions y lié est alors exécuté et le traitement de la commande est terminé.

Exemple

```
#include<stdio.h>
void main()
{   int A,B;
    printf(" Donner 2 entiers : ") ;
    scanf("%d %d",&A, &B) ;
    if(A>B)
        printf("%d est plus grand que %d\n",A,B) ;
    else
        if(A<B)
            printf("%d est plus petit que %d\n",A,B) ;
        else
            printf("%d est égal a %d\n",A,B) ;
}
```

Remarque :

En C une partie else est toujours liée au dernier if.

Exemple :

```
if(a<=b)
   if(b<=c)
       printf("Ordonné");
   else
       printf(" Non ordonné");
```

5. Les opérateurs conditionnels

Syntaxe :

<Cond> ? <expr 1> : <expr 2>;

1. Si <Cond> fournit une valeur différente de 0, alors la valeur de <expr 1> est fourni comme résultat.
2. Si <Cond> fournit la valeur 0, alors la valeur de <expr 2> est fourni comme résultat.

Exemple :

```
if(a>b)
   Max=a;
else
   Max=b;
```

Peut être remplacé par : Max=(a>b)?a:b;

6. Structure conditionnelle à choix multiples (switch)

Syntaxe :

switch (<expression>)

{ case constante 1 : [<instruction(s)>;] break;

case constante 2 : [<instruction(s)>;] break;

...

case constante N : [<instruction(s)>;] break;

[default : [<instruction(s)>;]]

}

1. Le mot clé "switch" doit être suivi d'une expression de type, entier, par exemple Le nom d'une variable de type correspondant ou une expression arithmétique.
2. Constante : expression constante entière.
3. Instruction(s) : séquence d'instructions quelconques.
4. Il est possible d'utiliser le mot-clé "default" comme étiquette à laquelle le programme se branchera dans le cas où aucune valeur satisfaisante n'aura été rencontrée auparavant.

N.B : Les crochets [ ] signifient que ce renferment est facultatif.

Exemple:

```
#include<stdio.h>
void main()
{   int NB ;
    printf(" Donner un entier : "); scanf("%d",&NB);
    switch(NB)
    {       case 0 : printf(" \n Nombre Zéro ");break;
            case 1 :
            case 3 :
            case 5 :
            case 7 :
            case 9 : printf(" \n Nombre impair ");break;
            case 2 :
            case 4 :
            case 6 :
            case 8 : printf(" \n Nombre pair ");break;
            default : printf(" \n Nombre erroné ");
    }
}
```