Programmation

En Langage C

Chaps 1 & 2

Cours TD TP

Sommaire

Partie II : Programmation C

Chap I : Introduction à la programmation C

- Phases de production d'un application C
- Structure d'un programme C
- Exemples de programmes C

Chap II : Les Concepts de base

- Types primitifs
- Déclarations de constantes et de variables
- Opérateurs
- Fonctions d'entrées sortie
- Les Structures de contrôle

I. Phases d'élaboration d'un programme C

L'élaboration d'un programme C passe par 4 phases :

1. Edition du code source

Cette phase consiste à écrire le code du programme C en utilisant un éditeur de texte. Ce code est enregistrer dans un fichier texte d'extension « .C ».

2. Traitement du pré-processeur

Consiste à préparer la phase de compilation en effectuant les **transformations textuelles** , dans le code source, suivantes :

- L'i**nclusion** de code source (fichiers en-tête (Header) d'extension « .h ») :

  #include

- La compilation conditionnelle : #if … #else #end if

- Le traitement de macros : #define

3. Compilation

Consiste à traduire le code généré par le pré-processeur en langage machine. Le résultat de compilation porte le nom de module objet (fichier.obj). Ce module objet n'est pas directement exécutable.

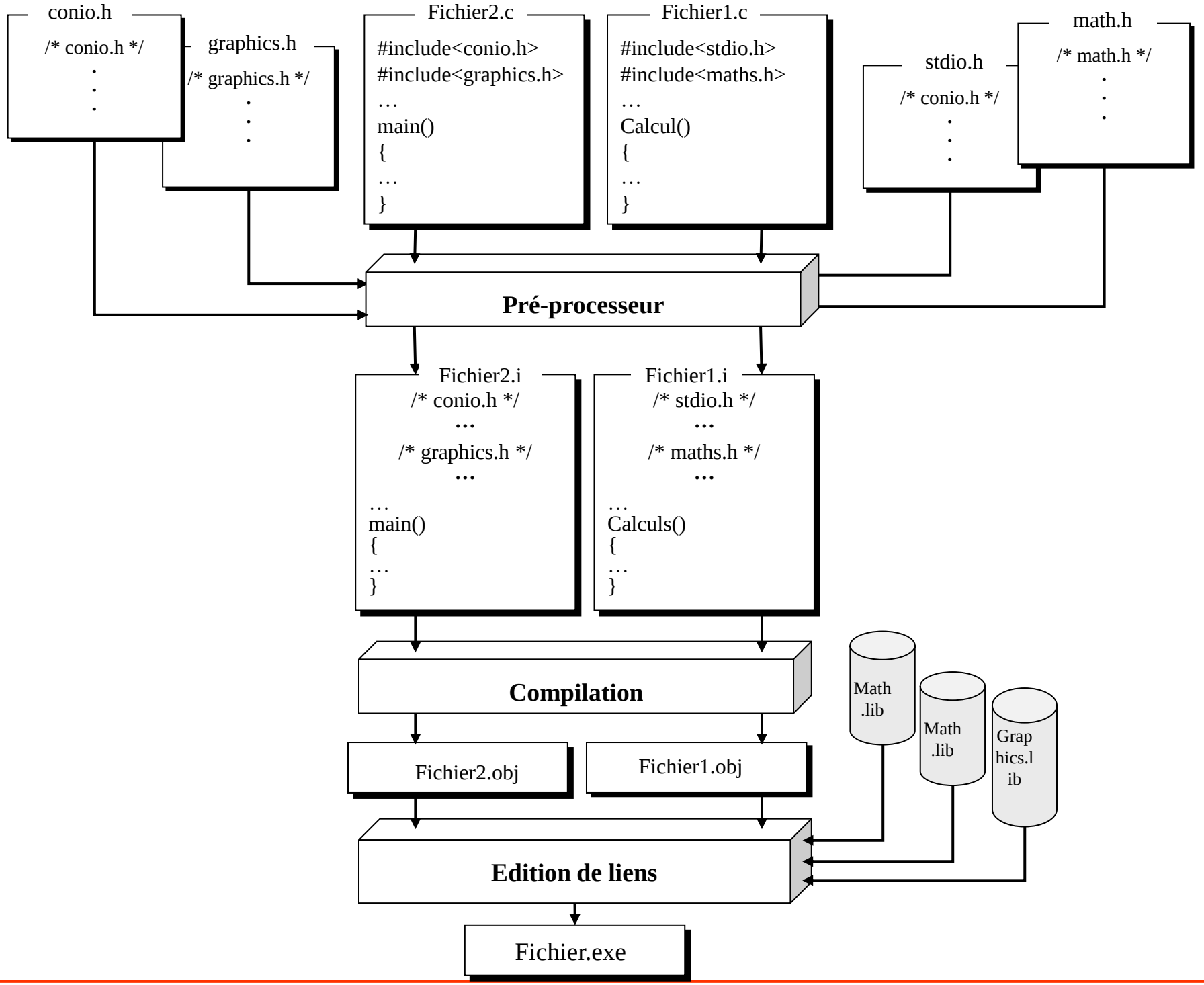
Le compilateur effectue deux opérations :

- Analyse (lexicale, syntaxique et sémantique)
- Synthèse (génération et optimisation du code objet)

4. Édition de liens

Produit, à partir d'un ou de plusieurs fichiers objets et des bibliothèques standards, un fichier exécutable (fichier.exe). Outre l'assemblage des divers fichiers objets, l'édition des liens inclut les définitions des fonctions prédéfinies utilisées par le programme.

Chap 1 : Introduction à la programmation C



II. Structure d'un programme C

Un programme C peut se décomposer par :

- Des **directives de compilation**
- Des **déclarations** de types, de variables et de constantes
- Des **prototypes de fonctions**
- Des **définitions de fonctions** parmi lesquelles la fonction **main**

```
/* Nom_du_programme */

/* Directives */
#include …
#define …

/* Déclarations */
types , Constantes et variables ;

/*Fonctions*/
prototypes_fonctions ;
```

```
/* definition_fonctions*/
  type nomfonction(args){
     ….
     }

/* Coprs du programme */
void main()
{
    declarations ;
    Action1 ;
    …}
```

III. Exemples de programmes C

Exemple 1 : Affichage des informations à l'écran

```
/* directive de compilation*/
#include <stdio.h>
#include <stdlib.h>
/* corps principal du */
int main(){
/* nettayage de l'écran */
system("cls");
/*affichage*/
puts ("Mini Système bancaire");
/* temporisation */
system("pause");
return 0;
}
```

Exemple 2 : Recherche du plus petit parmi trois nombres entiers notés n1 , n2 , n3.

```
/* Plus_Petit */
#include<stdio.h>
#include<stdlib.h>
/*Déclarations*/
int  n1 , n2 ,n3 , p;
Int min2(int x, int y); //prototype
/* fonction principale */
int main( ){
    system("cls");
    /* lecture de  n1, n2 et n3 */
    printf ("Introduire les nombres : " ) ;
   scanf ("%d  %d  %d",&n1,&n2,&n3);
    p =Min2(n1,n2);
    p =Min2(p,n3);
    /* phase 4 : afficher p */
    printf ("Le minimum = %d\n" , p) ;
    /* temporisation */
   system("pause"); return 0;
}
/*fonction minimum */
int Min2(int x , int y){
    if (x<y) return x;
    else return y;
}
```

Chap 2 : Les Concepts de base

I. Les types primitifs

Les entiers

Type	domaine	Taille(octet)
int	-32768 … 32767	2 | 4
unsigned int	0 … 65535	2 | 4
short	-32768 … 32767	2
unsigned short	0 … 65535	2
long	-2147483648 … 2147483647	4
unsigned long	0 … 4294967295	4

Les caractères

Type	domaine	Taille(octet)
char	-128 … 127	1
unsigned char	0 … 255	1

Remarque : En C, un caractère n'est autre que le nombre correspondant à son code **ASCII**. Par suite les types **char** et **unsigned char** peuvent être utilisés comme types entiers

Les réels

Type	Précision	Domaine en valeur absolue	Taille(octet)
float	5	$3.4 * 10^{-38}$ … $3.4\ 10^{38}$	4
double	15	$1.7 * 10^{-308}$ … $1.7 * 10^{308}$	8
Long double	19	$3.4 * 10^{-4932}$ … $1.1 * 10^{4932}$	10

Les booléens

il n'existe pas de type booléen en C. La valeur entière **0** est équivalente à la valeur **faux** et toute valeur **différente de 0** est équivalente à la valeur **vrai**.

On peut définir les macro-constantes **FALSE** et **TRUE** comme suite :

```
#define FALSE 0
#define TRUE 1
Ou bien :
enum bool {FALSE, TRUE  } ;
```

Les Chaînes

Il n'existe pas de type chaîne en C par contre on peut représenter un objet chaîne de caractère comme un tableau de caractères.

Exemples :

```
char NomClient[20], AdresseClient[60];
/* réserve deux blocs de 20 et 60 Octets resp. NomClient et AdresseClient
   désignent resp les adresses de ces deux blocs */

char *NomClient, *AdresseClient;
/* réserve deux cases nommées NomClient et AdresseClient de 2 Octets chacune
   ces deux cases servent à contenir les adresses de blocs mémoires à allouer */
```

II. Représentation des constantes

Un programme est composé d'un ensemble de **mots** et de **symboles** qui peuvent être :

- des **mots clés** (main, if, switch, do, while, for, …)
- des **identificateurs** d'objets variables, constantes et de fonctions
- des représentation de **constantes littérales**: ('A' , "chemin", 10, 23.25, …)
- des **symboles** : opérateurs (+, -, *, / , % …)

Constantes Entières

On distingue deux représentations :

- **Automatique** : dans ce cas C attribut à la constante le type le plus économique parmi les types entiers : int → long → unsigned long.

  Exemples 100, 2000 de type int

  52000 de type long

  4294967200 de type unsigned long

- **Type forcé** : Pour forcer un type entier imposé, on peut employer les suffixes u | U | l | L | ul | UL :

suffixe	Exemple	type
u ou **U**	550u	**unsigned (int ou long)**
l ou **L**	1234589L	**long**
ul ou **UL**	12092UL	**unsigned long**

Constantes Réelles

Les constantes réelles peuvent être indiquées :

- en notation **décimale** : 123.4 -0.001 1.0
- en notation **exponentielle** : **<+|-> <mantisse> e|E <exposant>**

  Avec

  <+|-> : le signe positif ou négatif du nombre

  <mantisse> : un décimal positif avec un seul chiffre devant la virgule

  <exposant> : un entier relatif

  Exemples : 1.234e-02 -1E-3 0.01E2

- Par défaut, les constantes réelles sont du type **double**. Pour forcer leur type, on utilise les suffixes f | F | l | L :

suffixe	Exemple	type
f ou **F**	5.335F	**float**
l ou **L**	3.14L	**long double**

Les constantes caractères

Les constantes caractères sont indiquées entre des apostrophes : par exemple 'a', 'b', 'A' …

Séquences d'échappement

Une séquence d'échappement est un couple de symboles dont le premier est le signe d'échappement '\'. Une séquence constitue un contrôle d'affichage ou d'impression :

\a	Sonnerie	**\r**	retour au début de ligne	\'	apostrophe
\b	curseur arrière	**\0**	NUL	\"	guillemets
\t	Tabulation	\\	trait oblique	**\f**	saut de page
\n	nouvelle ligne	**\?**	point d'interrogation	**\v**	tabulateur vertical

Les constantes chaînes de caractères

Les chaînes de caractères constantes sont indiquées entre guillemets " ".

Exemples

La chaîne de caractères vide est alors : ""

"Ce \ntexte \nsera réparti sur 3 lignes."

"Affichage de \"guillemets\"\n"

Remarques

'x' : Est un caractère constant codé en 1 octet

"x" : est une chaîne composée des deux caractères 'x' et '\0'. Elle est codée en 2 octets

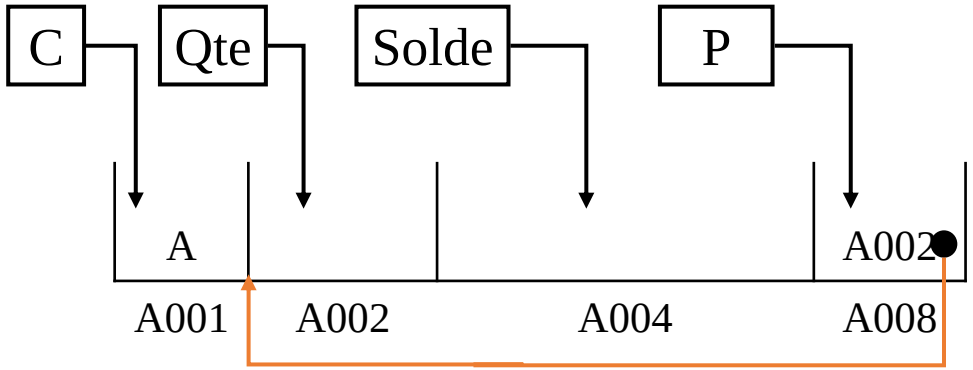
III. Les objets simples: Variables et constantes

Déclaration de variables simples

```
Nom_type  [*]nom_var1[=Valeur1] [,  [*]nom_var2[=valeur2], … ] ;
```

Exemples

```
char C    ='A';
int Qte    ;
float Solde ;
int *p=&Qte;
```

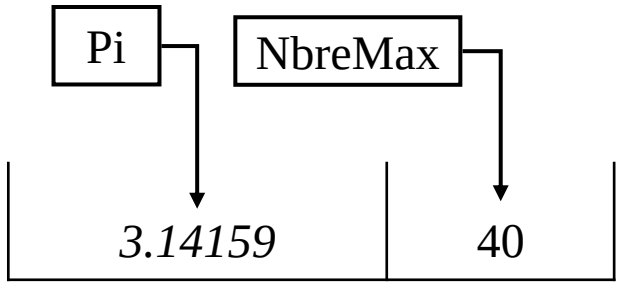


Déclaration de Constantes simples

```
const   type   nom_const1 = valeur  [, nom_const2  = valeur  ,…] ;
```

Exemples

```
const float Pi = 3.14159 ;
const int NbreMax    = 40 ;
```



IV. Les Opérateurs et expression

Action d'affectation

```
NomVar1 = Nomvar2 =  expression ;
```

Opérateurs arithmétiques

Opérateurs	Signification
+	Addition
–	Soustraction
*	Multiplication
/	Division
%	Reste de la division

Opérateurs relationnels

Opérateurs	Signification
>	Strictement supérieur
<	Strictement inférieur
>=	Supérieur ou égal
<=	Inférieur ou égal
= =	Egal
!=	Différent

Opérateurs logiques

Opérateurs	Noms
&&	ET Logique
||	OU logique
! (unaire)	Non logique

Exemples d'expressions

arithmétique	relationnelle	logique
b =1/(pow(sin(y),2) - pow(cos(y),2)) + 1	a == b	a = (x = = y) && z

Exemples d'expressions logiques

Soient A,B C et D des variables entières tel que :

A=10, C=-10 D=0 . Le tableau précise des expressions logiques ainsi que leurs valeurs.

Expression logique	Valeur
A = = (B=5)	0
(A && B || !C) && !D	1

Chap 2 : Les Concepts de base

Affectations élargies

Opérateur	Notation	équivalent	Opérateur	Notation	équivalent
+ =	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			

Opérateurs incrémentation

Opérateur	Notation		équivalent
	Préfixe	postfixe	
++	++x	x++	x = x + 1
--	--x	x--	x = x - 1

Autres Opérateurs

Nom	Opérateurs	Notations C
Opérateur conditionnel	? :	Condition ? expres_vrai : expres_faux
Opérateur de dimension	sizeof	Sizeof(type) | sizeof(variable)
Opérateur d'adressage	& (unaire)	&nom_variable
Opérateur de déréférencement	* (unaire)	*nom_pointeur
Opérateur de sélection	. ->	Nom_Structure.NomChamp Nom_Pointeur ->NomChamp
Opérateur de conversion	(type)	(int)f

Priorité des Opérateurs C

Priorité	Opérateur	Ordre
1 (la plus forte)	**( ) [] -> .**	→
2 (unaire)	**- ! ++ -- * & (cast) sizeof**	←
3	*** / %**	→
4	**+ -**	→
5	**<< >>**	→
6	**< <= > >=**	→
7	**== !=**	→
8	**&**	→
9	**^**	→
10	**|**	→
11	**&&**	→
12	**||**	→
13	**? :**	→
14 (la plus faible)	**= += -= *= /= %= &=**	←

IV. Les Entrées/Sorties

Ecriture des données

```
printf("Format" , expr [ , expr,…]) ; /*affiche des données selon un format */
puts(Chaine); /* affiche une chaîne de caractère */
putchar(caractere); /* affiche un caractère */
```

Spécificateurs de format pour printf

Symbole	*Type*
%d ou %i	int
%u	unsigned int
%hd ou %hi	short
%hu	unsigned short

Symbole	*Type*
%ld ou %lu	long
%lu	unsigned long
%c	char
%f ou %e	float /double

Symbole	*Type*
%Lf ou %Le	long double
%s	char* ou char [ ]

Lecture des données

```
scanf("Format" ,adres_var [ ,adresse_var,…]) ; /*Lit des donnée selon un format */
gets(Var_Chaine); /* Lit une chaîne de caractère */
var_caratere = getchar(); /* Lit un caractère */
var_caratere = getch() ; /* Lit un caratère sur le clavier */
```

Spécificateurs de format pour scanf

Identiques que pour printf à l'exception du type double

Symbole	*Type*
%f ou %e	float

Symbole	*Type*
%lf ou %le	double

Symbole	*Type*
%Lf ou %Le	long double

Exemple

Ecrire un programme qui permet de calculer et d'afficher la moyenne de 3 notes.

```
/*  Moyenne  */
#include<stdio.h>
#include<conio.h>

float Note1 , Note2 , Note3, Som, Moy ;
Void main()
{
    printf("Entrez trois réels : ");
    scanf("%f %f %f", &Note1,&Note2,&Note3) ;
    Som = Note1 ;
    Som += Note2 ;
    Som +=Note3 ;
    Moy = Som / 3 ;
    printf("La moyenne est de : %.2f", Moy) ;
    getch();
}
```

V. Les structures de contrôle

Bloc d 'actions

Séquence marquée par un début et une fin. Il est considéré comme une seule action. Des objets niveau bloc peuvent déclarés dans le bloc. Leur portée se limite au bloc.

```
{
    Declaration niveau bloc
    Action 1;
    Action 2;
    .....
    Action p;
}
```

Exemple

```
{
    int A,B ;
    scanf ("%d",&A);
    Calculer (A,&B);
    printf("A = %d  B = %d“,A,B) ;
}
```

Actions Conditionnelles

If(*Condition***)***BlocAction* ;
Ou bien
If(*Condition***)**
    BlocAction1 ;
else
    BlocAction2 ;

Imbrication de if

If(*Cond1***)**
    if(*Cond2***)** *Action21* ; **else** *Action22* ;
else
    Action12 **;**

Remarque

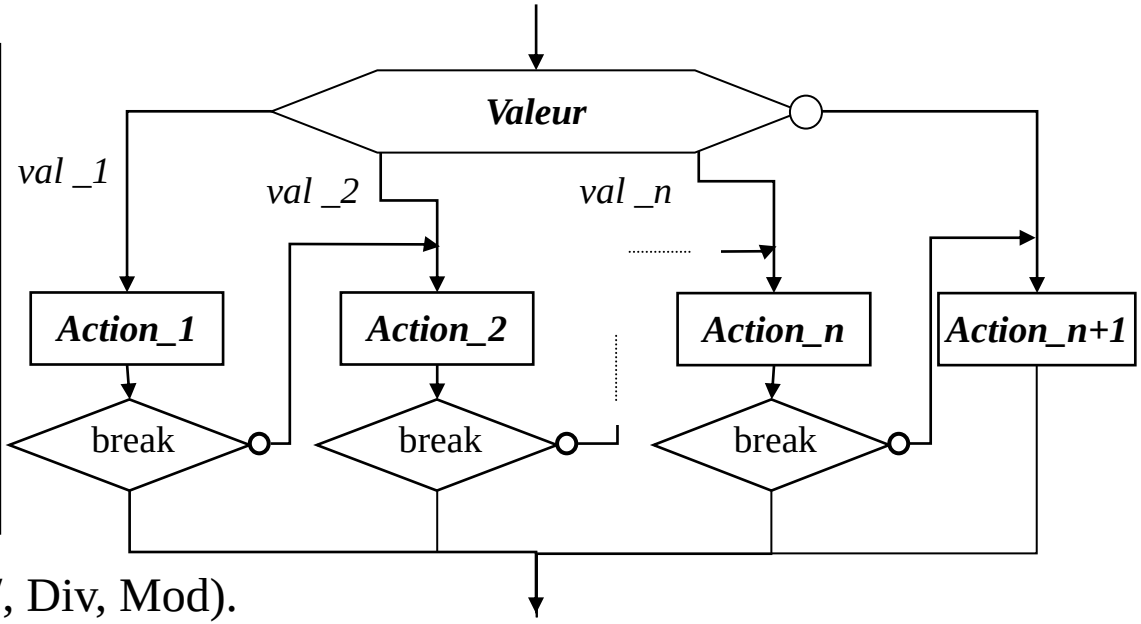
Dans une structure de if imbriqués, un else correspond toujours au if le plus proche

Exemple : Algorithme de résolution d'équation de second degré : $a\,x^2 + b\,x + c = 0$

```
/* Eq_Second_Degré */
#include<stdio.h>
#include<conio.h>
#include<math.h>
float a, b, c , Delta;
void main(){
      clrscr();
      printf("Donner les coefficients : ") ; scanf("%f %f %f", &a,&b,&c) ;
      if(a = =  0)
        if(b = = 0)
            if(c = =  0)printf("Infinités de solutions") ;else printf("Pas de solutions ");
        else
            printf("Equation de 1er degré, une racine réelle : %.4f ",-c/b) ;
      else{
        Delta = b*b –4*a*c ;
        if(Delta = = 0)
            printf("Une racine réelle double : %.4f " , -b/(2*a)) ;
        else
            if(Delta > 0)
                printf("Deux racines réelles :\n\tx1 = %.4f\n\tx2=%.4f", (-b +sqrt(Delta))/(2*a)    ,
                (-b +sqrt(Delta)) /(2*a)) ;
            else printf("Pas de solutions réelles") ;}getch();}
```

Action sélective

```
switch( valeur){
    case Val_1  : Action_1 ; [break];
    case Val_2  : Action_2 ; [break] ;
    …
    case Val_n : Action_n ;  [break] ;
    default      : Action_ n+1 ;
}
```



Exemple Calculatrice simplifiée. (+,-,*, /, Div, Mod).

```
/* Calculette */
#include<stdio.h>
#include<conio.h>
int Nbre1, Nbre2, CodeOper ;
void main(){
    /* Affichage du Menu */
    puts ("Calculatrice simplifiée : ") ;
    puts ("Addition                ……..  1 ");
    puts ("Soustraction            ……..  2 ");
    puts ("Produit                 ……..  3 ");
    puts ("Division exacte         ……..  4 ");
    puts ("Division entière        ……..  5 ");
```

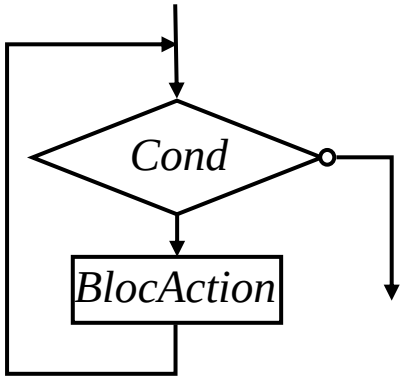
```
        /* Lecture de l'opération choisie  */
        printf ("Entrez le code de l''opération : ");
        scanf("%d", &CodeOper) ;
        /* Traitement du choix */
        if( (Codeoper>= 1) && (CodeOper <= 5)){
              printf("Entrez les valeurs des opérandes : ");
              scanf("%d %d", &Nbre1,&Nbre2);
              switch( CodeOper){
                    case  1 : printf("Le résultat est : %d " , Nbre1 + Nbre2) ;break;
                    case  2 : printf("Le résultat est :  %d" , Nbre1 - Nbre2) ;break;
                    case  3 : printf("Le résultat est : %d" , Nbre1 * Nbre2) ;break;
                    case  4 :
                       if(Nbre2 !=0) printf("Le résultat est : %f" , (float)Nbre1 / Nbre2) ;
                       else printf("Division par 0 impossible") ;
                       break;
                    case  5 :
                          if(Nbre2 != 0) printf("Le résultat est : %d" , Nbre1 / Nbre2);
                          else printf("Division par 0 impossible") ;
              }
        }
}
```

Chap 2 : Les Concepts de base

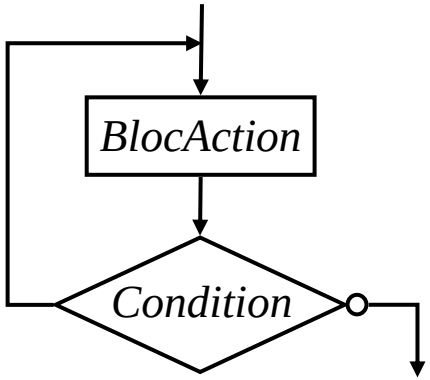
Actions répétitives (boucles)

En c, on distingue trois structure répétitives :

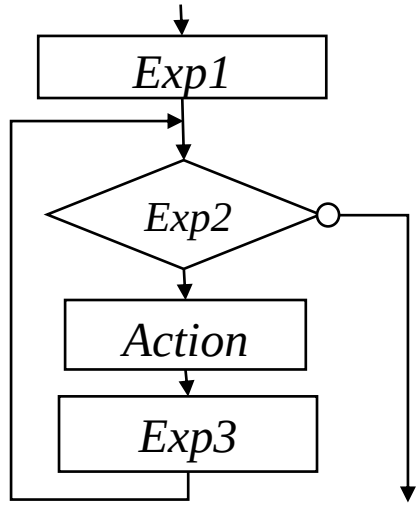
while (*Cond***)**
Action ;



do
BlocAction ;
while (*Cond***);**



For(*Exp1* **;** *Exp2* **;** *Exp3***)**
BlocAction;



Exemple1 : Masque de saisie d'un entier composé obligatoirement de 5 chiffres

```
/* Programme masque d'un Entier */
# include <stdio.h>
#include <conio.h>
#define  NbCh   5
long  Nombre ; int  I = 0 ; char  C ;
void main(){
    clrscr();
```

```
    do {
        C =getch() ;
        If (C >='0' && C<='9'){
            I++ ;putchar(c);
            Nombre = 10* Nombre + C-'0' ;}
    }while( I< NbCh );
    printf("\nle nombre est : %ld", Nombre);
    getch(); }
```

Exemple2 : calculer la moyenne d'une liste de notes dont le nombre n'est pas connu

```
/* Moyenne_Notes */
#include<stdio.h>
#include<conio.h>
float Note, Moy, Som ;
Int Nbre ;
void main(){
    /*initialisation*/ Som = 0; Nbre = 0 ;
    /*Saisie des notes et calcul de la somme des notes */
    printf("Entrez une note : ") ;scanf("%f",&Note) ;
    while(Note >= 0 && Note<=20){
        Som += Note ;
        Nbre ++ ;
        printf("Entrez une note : ") ;scanf("%f",&Note) ;
    }
    /*Calcul et affichage de la moyenne*/
    If(Nbre>0){
        Moy = Som / nbre ;
        printf("La moyenne des notes est : %.2f", Moy) ;}
    Else  printf("Liste de notes vide ") ;
    getch();}
```

Exemple3 : Calculer la somme des N premiers nombres premiers.

```
/*SomNbrePremiers */

#include<stdio.h>
#include<conio.h>
#include<math.h>

Int I , N , p=1 , j , Som=0 , EstPrem;

void main(){
     printf("N = "); scanf("%d",&N) ;
     for(I =1 ; I<=N ; I++){
          Som +=p ;
          do{  /* Rechercher le prochain nombre premier */
               EstPrem = 1 ; p++ ;
               for( j =2 ; j<=sqrt(p) ; j++) /* Examiner si p est premier */
                    if(p % j = = 0){
                         EstPrem = 0 ;
                         break ; /* Sortir de la boucle for */}
          }while(!EstPrem) ;
     }
     printf("La somme des %d premiers nombre Premiers est : %d ",N,Som) ;
     getch();
}
```

Instructions de rupture :

Instruction break

l'instruction **break** passe l'exécution à l'instruction suivant l'instruction **switch**, **while**, **do** et **for**.

Exemple

Sortie d'une boucle infinie :

```
/*saisie d'un caractère et affichage de son code ascii jusqu'à l'appui sur Echap */
#include<stdio.h>
#include<conio.h>
char c ;
void main(){
     clrscr() ;
     do{
          printf("tapez un caractère : ");
          c=getch();
          if(c==0){
               c=getch();
               printf("\ncaractère de code étendu : %d\n",c);}
          else{
               if(c==27) break;
               printf("\nle caractère %c a pour code %d \n",c,c);}
     } while(1);
}
```

Instruction continue

l'instruction **continue** passe l'exécution au début d'une boucle while, do ou for. Cela permet d'éviter certaines instruction d'une boucle

Exemple

Calculer et afficher la somme des entiers 1,2,4,6 et 7

```
/*Calculer la somme des entiers 1,2,4,6 et 7 */

#include<stdio.h>
#include<conio.h>

int s=0,i;
void main()
{
     clrscr() ;
     for(i=1;i<=7;i++){
         if(i==3 || i==5) continue;
         s+=i;
     }
     printf("La somme des entiers 1,2,4,6 et 7 = %d",s);
     getch();
}
```