

Atelier de Programmation 1 : Langage C

Sami ZGHAL
sami.zghal@planet.tn

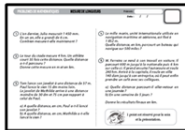
2012-2013

Plan

- 1 Introduction
- 2 Types & Variables & Opérateurs
- 3 Structures simples
- 4 Structures conditionnelles
- 5 Structures Itératives
- 6 Fonctions

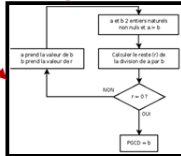
Introduction

Énoncé



Démarche

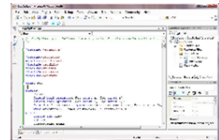
Algorithme



Traduction



Programme



Introduction

Énoncé

Une description détaillée d'un problème décrite dans un langage naturel

Algorithme

Un pseudo langage générique permettant de traiter des problèmes par concaténation d'instructions élémentaires

Introduction

Programme

Une suite d'opérations pré-déterminées destinées à être exécutées de manière automatique par un ordinateur en vue d'effectuer des travaux, des calculs arithmétiques ou logiques, ou simuler un déroulement

Langage de programmation

Un langage informatique permettant à un être humain d'écrire un code source qui est analysé par un ordinateur

Introduction

Cours : Programmation structurée

Langage de programmation : Langage C

Cours : Horaire

Cours : 1 heure 30 minutes par semaine

TD & TP : 1 heure 30 minutes par semaine

Langage C

Historique

- Mis au point par **D. Ritchie** et **B. W. Kernighan** au début des années 70
- Permettre de développer un langage qui permettrait d'obtenir un système d'exploitation de type UNIX portable

Atouts

- Langage C reste un des langages les plus utilisés actuellement
- Langage C est un langage comportant des instructions et des structures de haut niveau
- Principal intérêt du C est un langage très portable

Caractéristiques du langage C

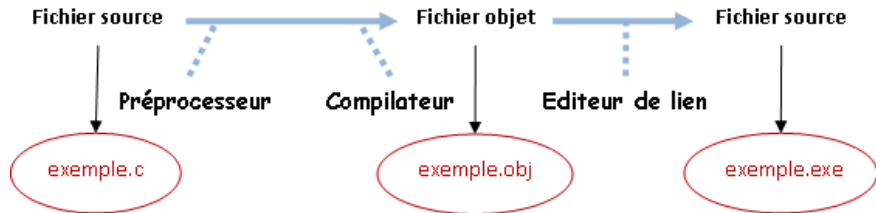
Fichier source

Un simple fichier texte dont l'extension est par convention .c

Exemple

```
#include <stdio.h>
int main()
{
    printf (" Ceci est votre premier programme\n" );
    return 0;
}
```


Compilation & Édition de lien



Préprocesseur

- Permet d'inclure dans le fichier source les éléments référencés par les instructions situées au début du fichier source

Compilation & Édition de lien

Compilateur

- Transforme le code source en code objet et le sauvegarde dans un **fichier objet**
- Traduit le fichier source en langage machine

Éditeur de lien

- Permet d'intégrer dans le fichier final tous les éléments annexes (fonctions ou librairies)
- Crée un fichier exécutable qui contient tout le nécessaire pour fonctionner de façon autonome

Plan

- 1 Introduction
- 2 Types & Variables & Opérateurs
- 3 Structures simples
- 4 Structures conditionnelles
- 5 Structures Itératives
- 6 Fonctions

Types & Variables & Opérateurs

- ① Types abstraits de données
- ② Variables
- ③ Opérateurs

Types abstraits de données

Types de données

Une spécification d'un ensemble de données et de l'ensemble des opérations qu'elles peuvent effectuer

Types abstraits de données

Respecte un cahier des charges qu'une structure de données doit implémenter

Types abstraits de données

Types de données prédéfinis en langage C

Type	Signification	Taille	valeurs
char	Caractère	1	-128 à 127
unsigned char	Caractère non signé	1	0 à 255
short int	Entier court	2	-32768 à 32767
unsigned short int	Entier court non signé	2	0 à 65535
int	Entier	2 (16 bits) 4 (32 bits)	-32768 à 32767 -2147483648 à 2147483647
unsigned int	Entier court non signé	2 (16 bits) 4 (32 bits)	0 à 65535 0 à 4294967295
long int	Entier long	4	-2147483648 à 2147483647
unsigned long int	Entier long non signé	4	0 à 4 294 967 295
float	Flottant (réel)	4	$3.4 \cdot 10^{-38}$ à $3.4 \cdot 10^{38}$
double	Flottant double	8	$1.7 \cdot 10^{-308}$ à $1.7 \cdot 10^{308}$
long double	Flottant double long	10	$3.4 \cdot 10^{-4932}$ à $3.4 \cdot 10^{4932}$

Types abstraits de données

Nombre entier

Un nombre sans virgule qui peut être exprimé dans différentes bases :

- **Base décimale** : une suite de chiffres unitaires (de 0 à 9) ne devant pas commencer par le chiffre 0
- **Base hexadécimale** : une suite d'unités (de 0 à 9 ou de A à F (ou a à f)) devant commencer par 0x ou 0X
- **Base octale** : une suite d'unités (incluant uniquement des chiffres de 0 à 7) devant commencer par 0

Types abstraits de données

Nombre à virgule flottante

Un nombre à virgule, il peut toutefois être représenté de différentes façons

- **Entier décimal** : 895
- **Nombre comportant un point** : 845.32
- **Fraction** : $27/11$
- **Nombre exponentiel** : $2.75e-2$, $35.8E+10$, $.25e-2$

Types abstraits de données

float

Codé sur 32 bits :

- **Mantisse** : 23 bits
- **Exposant** : 8 bits
- **Signe** : 1 bit

double

Codés sur 64 bits :

- **Mantisse** : 52 bits
- **Exposant** : 11 bits
- **Signe** : 1 bit

Types abstraits de données

long double

Codé sur 80 bits :

- **Mantisse** : 64 bits
- **Exposant** : 15 bits
- **Signe** : 1 bit

Nombres réels

- Précision des nombres réels est approchée
- Précision dépend du nombre de positions décimales :
 - 1 **float** : 6 chiffres
 - 2 **double** : 15 chiffres
 - 3 **long double** : 17 chiffres

Types abstraits de données

char

Permet de stocker la valeur ASCII d'un caractère

- Permet de stocker la valeur ASCII d'un caractère
- Permet de stocker une lettre : 'B'
- Permet de stocker un chiffre : 66

Variables

Définition

Objet repéré par son nom, pouvant contenir des données, pourront être modifiées lors de l'exécution du programme

En langage C

- Variable typée : données contenues dans la variable possèdent un type
- Stockée dans la mémoire : RAM
- Occupe un nombre d'octets : type de donnée stockée

Variables

Nom de la variable

- Peut être long : le compilateur ne prend en compte 32 premiers caractères
- Doit commencer par une lettre (majuscule ou minuscule) ou un " _ "
- Ne doit pas commencer par un chiffre
- Peut comporter des lettres, des chiffres et le caractère " _ "
- Ne peut pas comporter des espaces
- Ne peuvent pas être les mots réservés du langage

Variables

Nom de variable correct	Nom de variable incorrect	Raison
Variable	Nom de Variable	comporte des espaces
Nom_De_Variable	123Nom_De_Variable	commence par un chiffre
nom_de_variable	toto@mailcity.com	caractère spécial @
nom_de_variable_123	Nom-de-variable	signe - interdit
_707	goto	mot réservé

Variables

Déclarartion de variables

Définir une variable :

- Donner un nom à la variable
- Préciser le type de données à utiliser

Déclarartion de variables

- Une variable :
type Nom_de_la_variable;
- Plusieurs variables du même type :
type Nom_de_la_variable1, Nom_de_la_variable2, ...;

Opérateurs

- ① Opérateurs de calcul
- ② Opérateurs d'assignation
- ③ Opérateurs d'incrémentement
- ④ Opérateurs de comparaison
- ⑤ Opérateurs logiques

Opérateurs

Opérateurs de calcul

Permettent de modifier mathématiquement la valeur d'une variable

Opérateur	Dénomination	Effet	Exemple
+	Opérateur d'addition	Ajoute deux valeurs	$x+3$ (10)
-	Opérateur de soustraction	Soustrait deux valeurs	$x-3$ (4)
*	Opérateur de multiplication	Multiplie deux valeurs	$x*3$ (21)
/	Opérateur de division	Divise deux valeurs	$x/3$ (2.333)
=	Opérateur d'affectation	Affecte une valeur à une variable	$x=3$ (3)

Opérateurs

Opérateurs d'assignation

Permettent de simplifier des opérations d'ajout, de soustraction, de multiplication et division

Opérateur	Effet
<code>+=</code>	Additionne deux valeurs et stocke le résultat dans la variable
<code>-=</code>	Soustrait deux valeurs et stocke le résultat dans la variable
<code>*=</code>	Multiplie deux valeurs et stocke le résultat dans la variable
<code>/=</code>	Divise deux valeurs et stocke le résultat dans la variable

Opérateurs

Opérateurs d'incrémentation

Permettent d'augmenter ou diminuer d'une unité une variable

Opérateur	Dénomination	Exemple	Résultat (x=7)
++	Incrémentatation	x++	8
--	Décrémentatation	x--	6

Opérateurs

Opérateurs de comparaison

Permettent de comparer deux valeurs

Opérateur	Dénomination	Exemple	Résultat
==	Opérateur d'égalité	x==3	1 si x est égal à 3, sinon 0
<	Opérateur d'infériorité stricte	x<3	1 si x est inférieur à 3, sinon 0
<=	Opérateur d'infériorité	x<=3	1 si x est inférieur ou égal à 3, sinon 0
>	Opérateur de supériorité stricte	x>3	1 si x est supérieur à 3, sinon 0
>=	Opérateur de supériorité	x>=3	1 si x est supérieur ou égal à 3, sinon 0
!=	Opérateur de différence	x!=3	1 si x est différent de 3, sinon 0

Opérateurs

Opérateurs logique

Permettent de vérifier si plusieurs conditions sont vraies

Opérateur	Dénomination	Effet
	OU logique	Vérifie qu'une des conditions est réalisée $((C1) (C2))$
&&	ET logique	Vérifie que toutes les conditions sont réalisées $((C1) \&\& (C2))$
!	NON logique	Inverse l'état d'une variable booléenne $(!C)$

Opérateurs

Opérateurs logique

Permettent de de vérifier si plusieurs conditions sont vraies

C1	C2	C1 C2	C1&&C2	!C1
V	V	V	V	F
V	F	V	F	F
F	V	V	F	V
F	F	F	F	V

Plan

- 1 Introduction
- 2 Types & Variables & Opérateurs
- 3 Structures simples
- 4 Structures conditionnelles
- 5 Structures Itératives
- 6 Fonctions

Structures simples

- 1 Structure générale d'un programme
- 2 Affectation
- 3 Affichage sur écran
- 4 Lecture à partir du clavier
- 5 Exemple

Structure générale d'un programme

```
#include <stdio.h>
#define nbre 23
typedef

void main()
{
    int x;

    instruction_1;
    instruction_2;
    instruction_3;
    :
    instruction_n;
}
```

Affectation

Affectation ou désignation

- Permet d'attribuer une valeur à une variable
- Valeur doit être de même type (compatible) que la variable
- Syntaxe : `nomvariable = valeur;`

Exemple

```
int x,y,z;
```

- `x=2; y=3;`
- `x=y+2;`
- `x=x+2;`
- `w=x;`

Affectation

Exemple

```
int x; char alpha;
```

- `x=2;`
- `alpha='A';`
- `x=alpha;`

Exemple

```
float x,y; double s;
```

- `x=2.24;`
- `y=76.45;`
- `s=x+y;`

Affichage sur écran

`printf();`

- Permet d'afficher sans retour à la ligne
- Appartient à `stdio.h`
- Utilisation de **`printf`** nécessite d'inclure la bibliothèque `stdio.h`

Afficher un message

```
printf(" Bonjour" );
```

Affichage sur écran

Afficher un retour à la ligne

```
printf("\n");
```

Afficher une tabulation

```
printf("\t");
```

Affichage sur écran

`puts();`

- Permet d'afficher un message avec retour à la ligne
- Appartient à `stdio.h`
- Utilisation de **puts** nécessite d'inclure la bibliothèque `stdio.h`

Afficher un message

```
puts(" Bonjour" );
```

Affichage sur écran

Afficher le contenu d'une variable

- `printf("%format", nomvariable);`
- `format` : indique le type de variable
- `c` : variable de type caractère
- `d` : variable de type entier
- `f` : variable de type réel

Affichage sur écran

```
int x;  
char alpha;  
float w;  
  
x=2;  
printf("%d",x);  
alpha='A';  
printf("%c",alpha);  
w=2.23;  
printf("%f",w);
```


Affichage sur écran

Affichage mixte : message et variables

```
printf(" La somme de %d et %d est %d" ; x , y , s );
```

Lecture à partir du clavier

`scanf();`

- Permet effectuer une lecture formatée à partir du clavier
- Appartient à `stdio.h`
- Utilisation de **`scanf`** nécessite d'inclure la bibliothèque `stdio.h`

Lire le contenu d'une variable

- `scanf("%format",&nomvariable);`
- `format` : indique le type de variable
- `c` : variable de type caractère
- `d` : variable de type entier
- `f` : variable de type réel

Lecture à partir du clavier

```
int x
char alpha;
float w;

printf(" Saisir un entier : ");
scanf("%d",&x);

printf(" Saisir un caractère : ");
fflush(stdin);
scanf("%c",&alpha);

printf(" Saisir un reel : ");
scanf("%f",&w);
```

Lecture à partir du clavier

```
getch();
```

- Permet de lire un caractère à partir du clavier
- Appartient à `conio.h`
- Utilisation de **getch** nécessite d'inclure la bibliothèque `conio.h`

```
#include <conio.h>
void main()
{
    char alpha;

    alpha=getch();

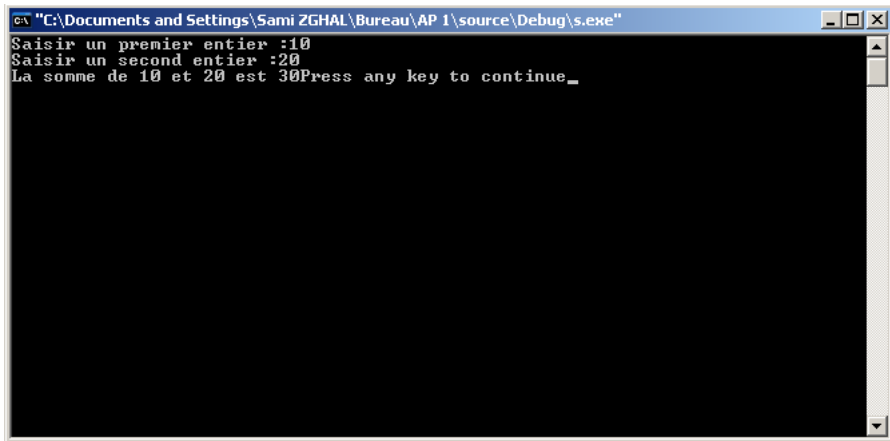
}
```

Exemple

```
#include <stdio.h>
void main()
{
    int x,y,s;

    printf(" Saisir un premier entier :");
    scanf("%d",&x);
    printf(" Saisir un second entier :");
    scanf("%d",&y);
    s=x+y;
    printf(" La somme de %d et %d est %d",x,y,s);
}
```

Exemple



A screenshot of a Windows command prompt window. The title bar reads "C:\Documents and Settings\Sami ZGHAL\Bureau\AP 1\source\Debug\s.exe". The command prompt shows the following text:

```
Saisir un premier entier :10  
Saisir un second entier :20  
La somme de 10 et 20 est 30Press any key to continue_
```

Plan

- 1 Introduction
- 2 Types & Variables & Opérateurs
- 3 Structures simples
- 4 Structures conditionnelles
- 5 Structures Itératives
- 6 Fonctions

Introduction

- ① Définition
- ② Structure conditionnelle à 2 choix
- ③ Structure conditionnelle à choix multiple

Définition

Structure conditionnelle

Instruction permettant d'effectuer des traitement en fonction d'une certaine condition

Condition

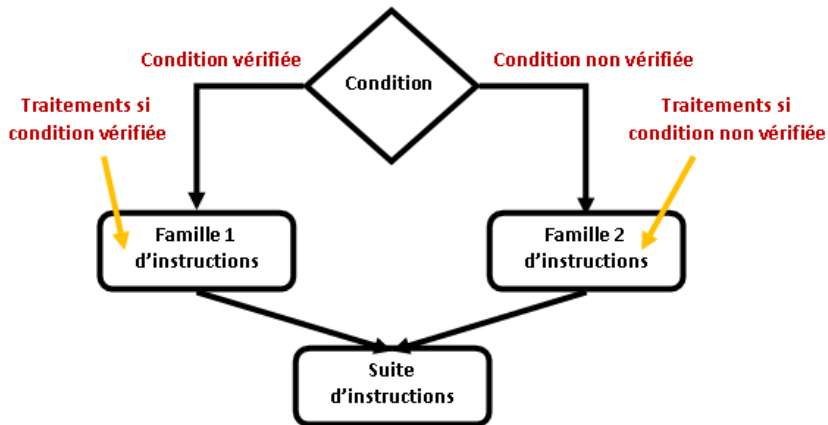
- **Condition simple** : condition contenant un opérateur de comparaison (`==`, `<`, `<=`, `>`, `>=` et `!=`)
- **Condition composée** : plusieurs conditions simple combinées avec opérateurs logiques (`||`, `&&` et `!`)

Structure conditionnelle à 2 choix

Définition

- Instruction conditionnelle à 2 choix permet d'exécuter deux familles d'instructions en fonction d'une condition
- Famille 1 d'instructions : exécutée quand **la condition est vérifiée**
- Famille 2 d'instructions : exécutée quand **la condition n'est pas vérifiée**

Structure conditionnelle à 2 choix



Structure conditionnelle à 2 choix

Syntaxe

```
if (condition)
{
    l_1 ;
    :
    l_n ;
}
else
{
    Inst_1 ;
    :
    Inst_m ;
}
```

Structure conditionnelle à 2 choix

Exemple

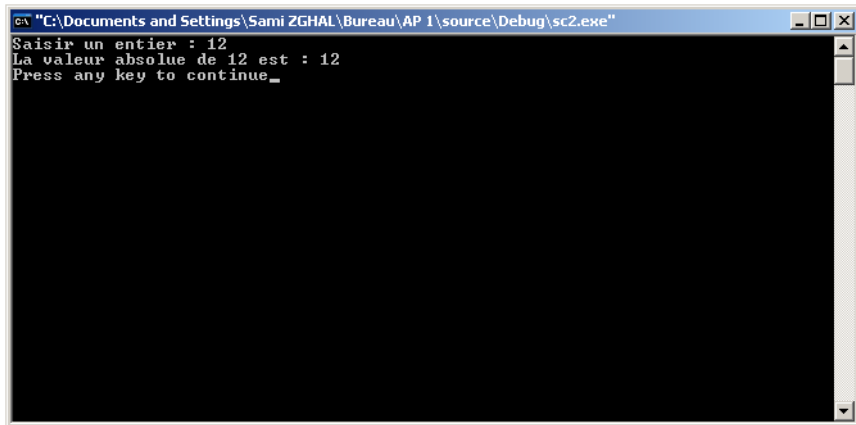
Écrire un programme qui permet de déterminer la valeur absolue d'un entier

Programme

```
#include <stdio.h>
void main()
{
    int x, abs;

    printf(" Saisir un entier : ");
    scanf("%d",&x);
    if (x>=0)
    { abs=x; }
    else
    { abs=-x; }
```

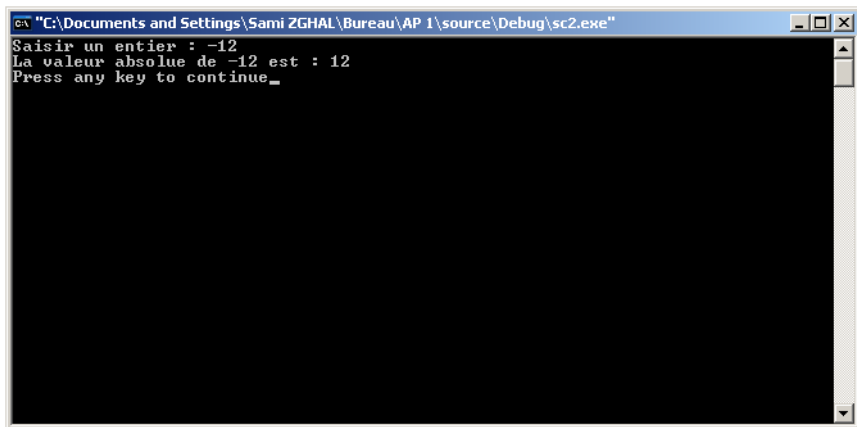
Structure conditionnelle à 2 choix



A screenshot of a Windows command prompt window. The title bar shows the file path: "C:\Documents and Settings\Sami ZGHAL\Bureau\AP 1\source\Debug\sc2.exe". The window contains the following text: "Saisir un entier : 12", "La valeur absolue de 12 est : 12", and "Press any key to continue_". The text is displayed in a monospaced font on a black background.

```
G:\ "C:\Documents and Settings\Sami ZGHAL\Bureau\AP 1\source\Debug\sc2.exe"  
Saisir un entier : 12  
La valeur absolue de 12 est : 12  
Press any key to continue_
```

Structure conditionnelle à 2 choix



A screenshot of a Windows command prompt window. The title bar shows the file path: "C:\Documents and Settings\Sami ZGHAL\Bureau\AP 1\source\Debug\sc2.exe". The window contains the following text:

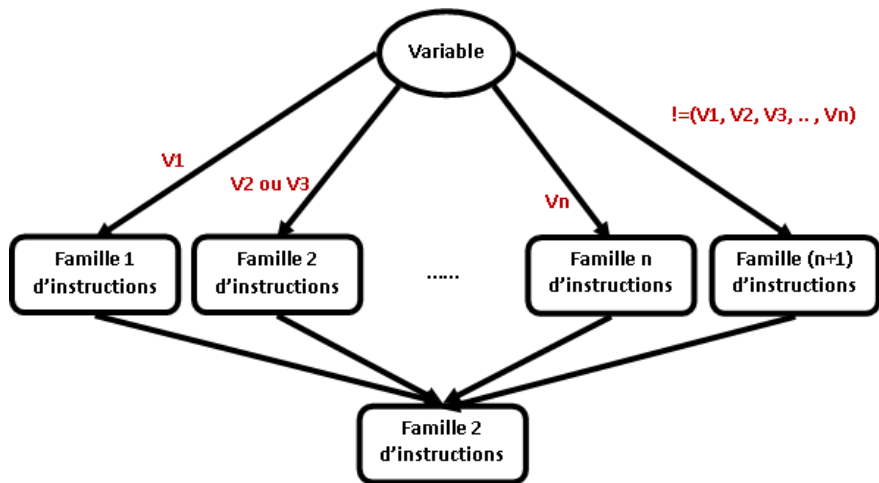
```
Saisir un entier : -12
La valeur absolue de -12 est : 12
Press any key to continue_
```

Structure conditionnelle à choix multiple

Définition

- Instruction conditionnelle à choix multiple permet d'exécuter plusieurs familles d'instructions en fonction du contenu d'une variable
- Famille 1 d'instructions : exécutée quand **la variable contient la valeur V1**
- Famille 2 d'instructions : exécutée quand **la variable contient la valeur V2**
- .
- .
- Famille n d'instructions : exécutée quand **la variable contient la valeur Vn**

Structure conditionnelle à choix multiple



Structure conditionnelle à choix multiple

Syntaxe

```
switch (Variable)
{
    case V1:
        Famille 1 d'instructions;
        break;

    case V2, V3:
        Famille 2 d'instructions;
        break;

    :
    case Vn:
        Famille n d'instructions;
        break;

    default:
        Famille (n+1) d'instructions;
}
```

Structure conditionnelle à choix multiple

Exemple

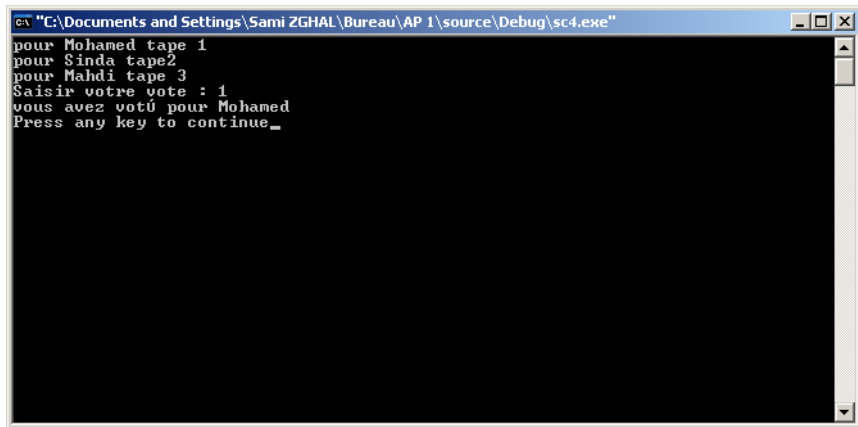
Écrire un programme qui permet de voter pour une des trois personnes (1:Mohamed, 2:Sinda et 3:Mahdi) et d'afficher le prénom de la personne pour laquelle vous avez voté

Structure conditionnelle à choix multiple

Programme

```
#include <stdio.h>
int main(void)
{
    int i;
    printf("pour Mohamed tape 1\n");
    printf("pour Sinda tape 2\n");
    printf("pour Mahdi tape 3\n");
    printf("Saisir votre vote : ");
    scanf("%d",&i);
    switch(i)
    {
        case 1: printf("vous avez voté pour Mohamed\n");
                break;
        case 2: printf("vous avez voté pour Sinda\n");
                break;
        case 3: printf("vous avez voté pour Mahdi\n");
                break;
```

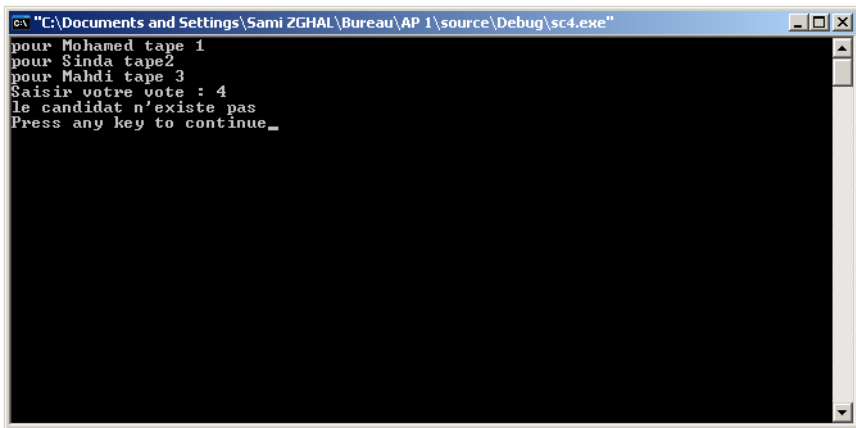
Structure conditionnelle à choix multiple



A screenshot of a Windows command prompt window. The title bar reads "C:\ "C:\Documents and Settings\Sami ZGHAL\Bureau\AP 1\source\Debug\sc4.exe"". The window contains the following text:

```
pour Mohamed tape 1  
pour Sinda tape2  
pour Mahdi tape 3  
Saisir votre vote : 1  
vous avez voté pour Mohamed  
Press any key to continue_
```

Structure conditionnelle à choix multiple



A screenshot of a Windows command prompt window. The title bar reads "C:\Documents and Settings\Sami ZGHAL\Bureau\AP 1\source\Debug\sc4.exe". The command prompt shows the following text:

```
pour Mohamed tape 1  
pour Sinda tape2  
pour Mahdi tape 3  
Saisir votre vote : 4  
le candidat n'existe pas  
Press any key to continue_
```

Plan

- 1 Introduction
- 2 Types & Variables & Opérateurs
- 3 Structures simples
- 4 Structures conditionnelles
- 5 Structures Itératives
- 6 Fonctions

Introduction

- ① Définition
- ② Boucle **for**
- ③ Instruction **while**
- ④ Instruction **do .. while**
- ⑤ Types composés

Définition

Structure itérative

Instruction permettant d'effectuer des traitements qui se répètent un certain nombre (connu ou inconnu) de fois

Boucle **for**

Syntaxe

```
for (initialisation; condition de progression; avancement)
{
    liste d'instructions;
}
```

for

- Initialisation : point de départ (compteur = valeur initiale)
- Condition de progression : assure la progression et l'arrêt (compteur <, <=, >, >= valeur finale)
- Avancement : incrémentation (compteur=compteur +,- valeur)

Boucle for

Exemple

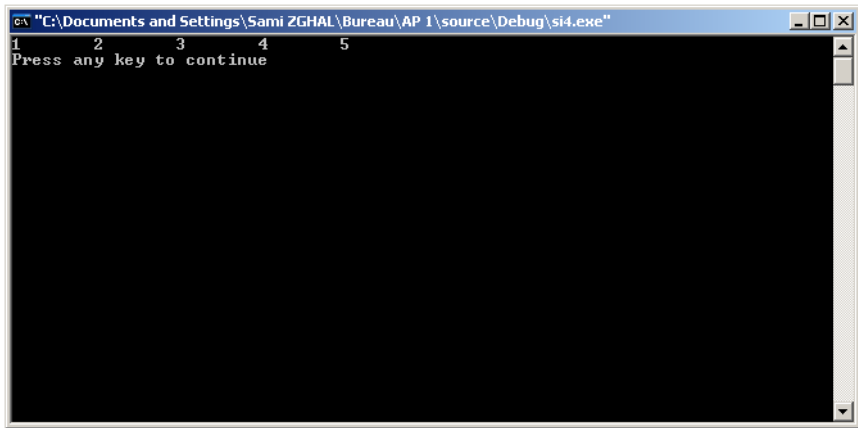
Écrire un programme qui permet d'afficher sur écran les entiers de 1 à 5

Programme

```
#include <stdio.h>
void main()
{
    int i;

    for (i=1; i<6; i++)
    { printf("%d\t", i); }
    printf("\n");
}
```

Boucle for



A screenshot of a Windows command prompt window. The title bar reads "C:\ "C:\Documents and Settings\Sami ZGHAL\Bureau\AP 1\source\Debug\si4.exe"". The window content shows the output of a C program. The first line displays the numbers 1 through 5, each centered under a column. The second line displays the text "Press any key to continue".

```
C:\ "C:\Documents and Settings\Sami ZGHAL\Bureau\AP 1\source\Debug\si4.exe"  
1      2      3      4      5  
Press any key to continue
```

Boucle **for**

Syntaxe

- Cette boucle affiche 5 fois la valeur de i (1, 2, 3, 4, 5)
- Elle commence à $i=1$, vérifie que i est bien inférieur à 6, etc. jusqu'à atteindre la valeur $i=6$, pour laquelle la condition ne sera plus réalisée, la boucle s'interrompt et le programme continuera son cours

Boucle **for**

Nombre de répétitions

- `for(i=0;i<10;i++)` exécute 10 fois la boucle (i de 0 à 9)
- `for(i=0;i<=10;i++)` exécute 11 fois la boucle (i de 0 à 10)
- `for(i=1;i<10;i++)` exécute 9 fois la boucle (i de 1 à 9)
- `for(i=1;i<=10;i++)` exécute 10 fois la boucle (i de 1 à 10)

Instruction **while**

Syntaxe

```
Initialisation ;  
while ( Condition )  
{  
    Liste d'instruction ;  
    Avancement ;  
}
```

Rôle

- **while** permet de répéter la liste d'instruction et l'avancement tant que la condition est vérifiée
- Nombre de répétitions non connu d'avance (avant l'exécution de l'instruction)
- Nombre de répétitions : au minimum un 0 ou plusieurs fois

Instruction **while**

while

- Initialisation : instruction attribuant une valeur à la variable de la condition
- Condition : condition d'avancement
- Avancement : instruction assurant la progression avec la condition

Instruction **while**

Exemple

On vous propose un travail dangereux. Le salaire quotidien commence à 1 millime et est multiplié par 2 tous les jours. Ainsi, vous recevrez 1 millime le premier jour, 2 millimes le deuxième jour, 4 millimes le troisième jour, et ainsi de suite. Vous voulez savoir combien de jours faut-il travailler pour devenir millionnaire en dinars. Ecrire un programme qui permet de le déterminer.

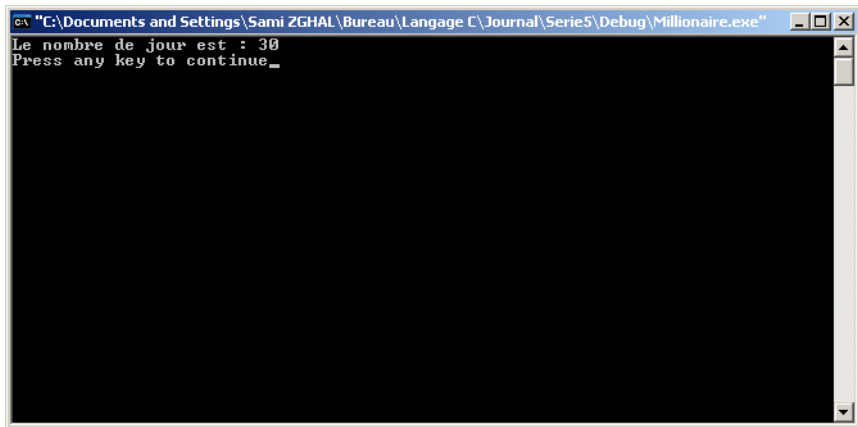
Instruction **while**

Programme

```
#include <stdio.h>
void main()
{
    int jour;
    float salaire , gain;

    salaire=1;
    jour=1;
    gain=salaire;
    while (gain <=1000000000)
    {
        salaire=salaire*2;
        gain=gain+salaire;
        jour++;
    }
    printf(" Le nombre de jour est : %d\n", jour);
}
```

Instruction **while**



```
C:\Documents and Settings\Sami ZGHAL\Bureau\Langage C\Journal\Serie5\Debug\Millionaire.exe
Le nombre de jour est : 30
Press any key to continue_
```

Instruction **do .. while**

Syntaxe

```
do
{
    Liste des instructions ;
}while ( Condition );
```

Rôle

- **do ..while** permet de répéter la liste d'instruction tant que la condition est vérifiée
- Nombre de répétitions non connu d'avance (avant l'exécution de l'instruction)
- Nombre de répétitions : au minimum 1 ou plusieurs fois

Instruction **do .. while**

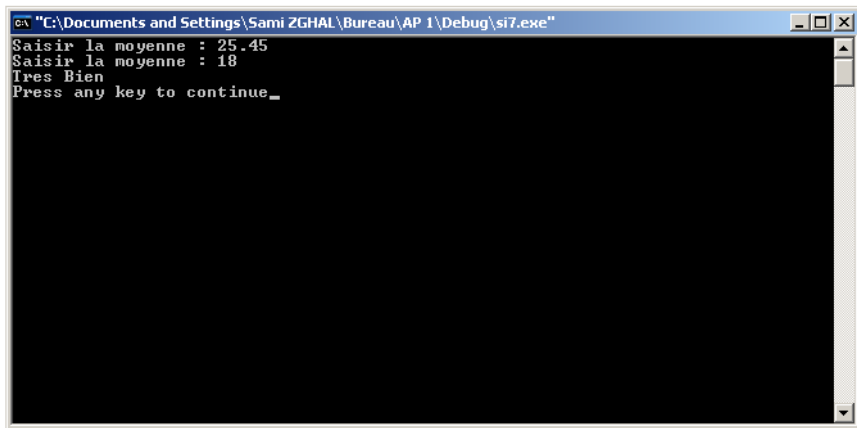
Exemple

écrire un programme qui permet de saisir une moyenne (comprise entre 0 et 20) et d'afficher la mention

Instruction `do .. while`

```
#include<stdio.h>
void main()
{
    float moyenne;
    do
    {
        printf(" Saisir la moyenne : " );
        scanf("%f",&moyenne);
    } while ((moyenne<0)|| (moyenne>20));
    if(moyenne<10)
    { printf(" Echec\n" );}
    else
    {
        if(moyenne<12)
        { printf(" Passable\n" );}
        else
        {
            if(moyenne<14)
            { printf(" Assez Bien\n" );}
            else
            {
                if(moyenne<16)
                { printf(" Bien\n" );}
                else
                { printf(" Tres Bien\n" );}
            }
        }
    }
}
```

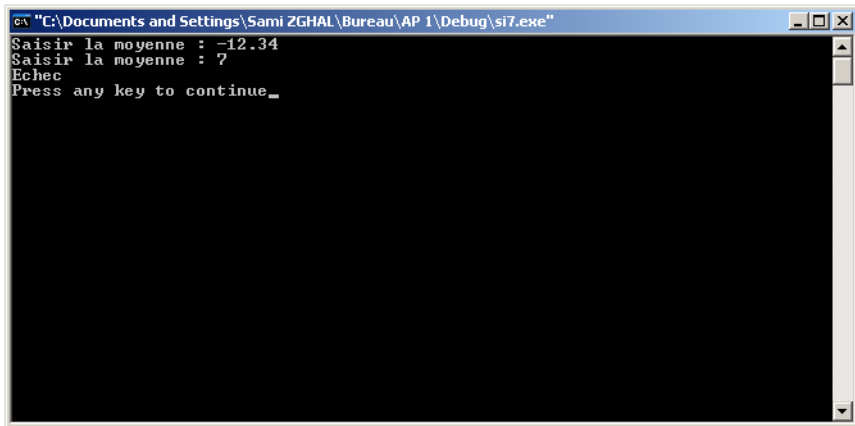
Instruction `do .. while`



A screenshot of a Windows command prompt window. The title bar reads "C:\Documents and Settings\Sami ZGHAL\Bureau\AP 1\Debug\si7.exe". The command prompt shows the following text: "Saisir la moyenne : 25.45", "Saisir la moyenne : 18", "Tres Bien", and "Press any key to continue_". The cursor is positioned at the end of the last line.

```
C:\Documents and Settings\Sami ZGHAL\Bureau\AP 1\Debug\si7.exe
Saisir la moyenne : 25.45
Saisir la moyenne : 18
Tres Bien
Press any key to continue_
```

Instruction `do .. while`



```
C:\Documents and Settings\Sami ZGHAL\Bureau\AP 1\Debug\si7.exe
Saisir la moyenne : -12.34
Saisir la moyenne : 7
Echec
Press any key to continue_
```


Types composés

- ① Tableau à une dimension : vecteur
- ② Tableau à 2 dimensions : matrice
- ③ Parcours total
- ④ Parcours partiel
- ⑤ Chaîne de caractères

Tableau à une dimension : vecteur

Définition

Variable de type tableau : variable permettant de stocker plusieurs valeurs de même type

Caractéristiques

- Nom de variable
- Type des éléments
- Nombre d'éléments

Déclaration & exemple

- `Type NomTableau[NombreCases];`
- `int T[10]; char T1[5]; float T2[6];`

Tableau à une dimension : vecteur

Caractéristiques d'une case

- Position (indice) : 0 à Nombre d'éléments - 1
- Valeur : contenu de la cas

Exemple

12	23	10	34	56	24
0	1	2	3	4	5

- `int T[6];`
- Case 0 : valeur 12
- Case 4 : valeur 56

Tableau à une dimension : vecteur

Remplir le tableau

```
printf("Remplir le tableau\n");  
for (i=0; i<=9; i++)  
{  
    printf("T[%d]=" , i);  
    scanf("%d",&T[i]);  
}
```

Afficher le contenu du tableau

```
printf("Le contenu du tableau est :\n");  
for (i=0; i<=9; i++)  
{  
    printf("%d\t", T[i]);  
}
```

Tableau à une dimension : vecteur

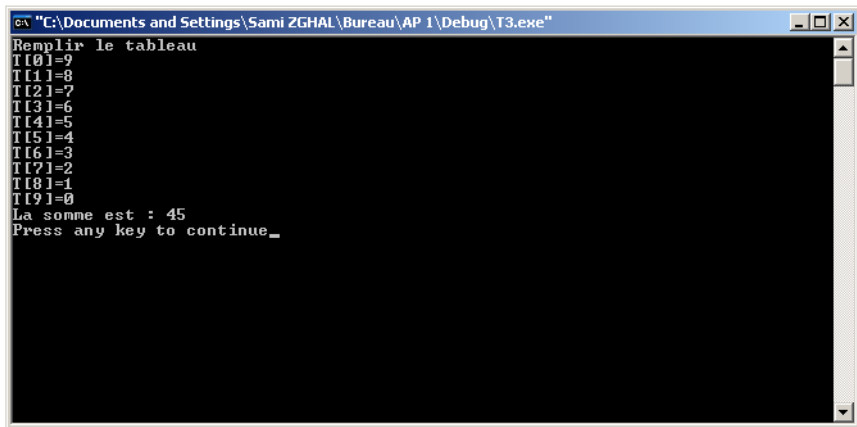
Exemple

Ecrire un programme qui permet d'additionner les éléments d'un tableau d'entiers composé de 10 cases

Tableau à une dimension : vecteur

```
#include <stdio.h>
void main()
{
    int T[10];
    int i,s;
    printf(" Remplir le tableau\n");
    for(i=0;i<10;i++)
    {
        printf("T[%d]=" , i );
        scanf("%d",&T[i]);
    }
    s=0;
    for(i=0;i<10;i++)
    {s=s+T[i];}
    printf(" La somme est : %d\n",s);
}
```

Tableau à une dimension : vecteur



```

C:\ "C:\Documents and Settings\Sami ZGHAL\Bureau\AP 1\Debug\T3.exe"
Remplir le tableau
T[0]=9
T[1]=8
T[2]=7
T[3]=6
T[4]=5
T[5]=4
T[6]=3
T[7]=2
T[8]=1
T[9]=0
La somme est : 45
Press any key to continue_

```

Tableau à 2 dimensions : matrice

Caractéristiques

- Nom de variable
- Type des éléments
- Nombre de lignes
- Nombre de colonnes

Déclaration & exemple

- Type NomMatrice[NombreLignes][NombreColonnes];
- `int M[10][3]; char M1[5][7]; float M2[6][3];`

Tableau à 2 dimensions : matrice

Caractéristiques d'une case

- Position (coordonnées):
 - 1 Numéro de la ligne : 0 à Nombre de lignes - 1
 - 2 Numéro de la colonne : 0 à Nombre de colonnes - 1
- Valeur : contenu de la cas

Tableau à 2 dimensions : matrice

Exemple

	0	1	2
0	12	23	34
1	45	56	67
2	78	90	4
3	26	71	5
4	92	7	6
5	10	19	17

- `int M[6][3];`
- Case (0,0) : valeur 12

Tableau à 2 dimensions : matrice

Remplir la matrice

```
printf("Remplir la matrice\n");  
for (i=0; i<=5; i++)  
{  
    //parcourir les lignes  
    for (j=0; j<=2; j++)  
    {  
        //parcourir les colonnes  
        printf("M[%d][%d]=" , i , j );  
        scanf("%d",&M[i][j]);  
    }  
    printf("\n");  
}
```

Tableau à 2 dimensions : matrice

Afficher le contenu de la Matrice

```
printf("Le contenu de la matrice est : \n");  
for (i=0; i<=5; i++)  
{  
    //parcourir les lignes  
    for (j=0; j<=2; j++)  
    {  
        //parcourir les colonnes  
        printf("%d", M[i][j]);  
    }  
    printf("\n");  
}
```

Tableau à 2 dimensions : matrice

Exemple

Ecrire un programme qui permet d'additionner les éléments d'une matrice d'entiers composée de 6 lignes et 3 colonnes

Tableau à 2 dimensions : matrice

```
#include<stdio.h>
void main()
{
    int M[6][3];
    int i,j,s;
    printf(" Le contenu de la matrice est : \n" );
    for ( i=0;i <=5;i++)
    {
        for (j=0;j <=2;j++)
        {
            printf("M[%d][%d]=" , i , j );
            scanf("%d",&M[i][j]);
        }
        printf("\n");
    }
    s=0;
    for ( i=0;i <=5;i++)
    {
        for (j=0;j <=2;j++)
        {s=s+M[i][j];}
    }
    printf(" La somme est : %d\n" ,s);
}
```

Tableau à 2 dimensions : matrice

C:\ "C:\Documents and Settings\Sami ZGHAL\Bureau\AP 1\Debug\M3.exe"

Le contenu de la matrice est :

M[0][0]=1

M[0][1]=2

M[0][2]=3

M[1][0]=4

M[1][1]=5

M[1][2]=6

M[2][0]=7

M[2][1]=8

M[2][2]=9

M[3][0]=10

M[3][1]=11

M[3][2]=12

M[4][0]=13

M[4][1]=14

M[4][2]=15

M[5][0]=16

M[5][1]=17

M[5][2]=18

La somme est : 171

Press any key to continue_

Parcours total

Exemple

Ecrire un programme qui compte les occurrences d'une lettre dans un tableau de caractères. Nombre d'occurrences = Nombre de fois que la lettre se trouve dans le tableau.

Parcours total

```
#include <stdio.h>
void main()
{
    char T[10];
    int i,n;
    char valeur;
    printf(" Remplir le tableau :\n");
    for(i=0;i<=9;i++)
    {
        printf("T[%d]=" , i);
        fflush(stdin); scanf("%c",&T[i]);
    }
    printf(" Saisir la valeur a rechercher : ");
    fflush(stdin);
    scanf("%c",&valeur);
    n=0;
    for(i=0;i<=9;i++)
    {
        if(T[i]==valeur){n++;}}
    printf(" Le nombre d'occurrence de %c est : %d\n",valeur,n);
}
```

Parcours partiel

Exemple

Ecrire un programme qui recherche dans un tableau d'entiers l'indice d'une valeur donnée (la première occurrence apparue).

Parcours partiel

```
i=0;
trouve=0;
while ((i<=9)&&(!trouve))
{
    if (T[i]==valeur)
    {
        trouve=1;
        indice=i;
    }
    else
    { i++; }
}
if (trouve)
{ printf("La premiere occurence de  %d se trouve
%d\n",valeur ,indice); }
else
{ printf("La valeur %d ne se trouve pas dans le tableau\n",valeur); }
```

Chaîne de caractères

Plan

- 1 Introduction
- 2 Types & Variables & Opérateurs
- 3 Structures simples
- 4 Structures conditionnelles
- 5 Structures Itératives
- 6 Fonctions

Fonctions