

## Structures de contrôle

### Instructions et blocs

- Une expression suivie d'un point virgule devient une instruction

`x = 0; i++; printf();`

sont des instructions valides.

`25; (3+5)*4;`

le sont également mais sont moins utiles

- `{` et `}` regroupent des déclarations et des instructions pour former une instruction unique.

`{ int i; for(i=0; i<10; ++i) t[i] = 0; }`

## Instructions if ... else (1)

- Syntaxe
  - `if(expression)`  
    `instruction;`
  - `if(expression)`  
    `instruction1`  
    `else`  
        `instruction2`
- Si l'expression est vraie, l'instruction 1 est exécutée, sinon l'instruction 2 est exécutée, si elle existe
- Exemple :
 

```
if(a>b){
    q = z;
    x = y [s];
}
else{
    printf("erreur\n");
    exit(1).
}
```

## Instructions if ... else (2)

- Lorsqu'on a plusieurs cas à tester, on peut enchaîner les `if ... else` :
 

```
if(expression)
    instruction
else
    if(expression)
        instruction
    else
        ...
```
- Operateur ternaire `? ... : ...`  
`(expression) ? valeur1 : valeur2`  
`max = (a>b) ? a : b;`

## switch ... case

- L'instruction switch ... case sert à traiter des choix multiples en fonction de la valeur d'une expression entière

- Syntaxe :

```
switch(expression){
    case val1 : instructions;
                break;
    case val2 : instructions;
                break;
    ...
    default : instructions;
}
```

- Exemple :

```
while((c=getchar())!=EOF){
    switch(c){
        case 'b' : printf("Bonjour");
                    break;
        case 'q' : printf("fin\n");
                    exit(0);
        default:
    }
}
```

## while

- Syntaxe :

```
while(expression)
    instruction
```

- Répète l'instruction tant que expression est vraie.

- Exemples :

```
while((c=getchar())!=EOF)
    putchar(c);
```

```
while((c=getchar())==' ');
```

## for (1)

- Syntaxe :  

```
for(expr1 ;expr2 ;expr3)
    instruction
```
- Sémantiquement équivalente à l'instruction while :  

```
expr1;
while(expr2){
    instruction
    expr3;
}
```
- Exemple :  

```
for(i=0 ;i<10 ;++i)
    t[i] = 0;
```

MIAGE - Université de Bordeaux

7

## for (2)

- On peut utiliser l'opérateur « , » pour exécuter plusieurs instructions simples dans l'initialiseur ou l'itérateur d'une boucle.
- Exemple : calcul de n! sur une seule ligne  

```
int i,f,n;
for(i=n, f=1;i>1; i--)
    f *= i;
```

MIAGE - Université de Bordeaux

8

## do ... while

- Syntaxe :  

```
do
    instruction
while(expression);
```
- L'instruction est d'abord exécutée, puis l'expression est évaluée. Si elle est vraie, on reboucle sur l'exécution de l'instruction.
- Exemple :  

```
do{
    depiler();
}while(!est_vide());
```
- Remarque : à la différence de while, l'instruction est toujours exécutée au moins une fois.

## break

- Sert à quitter la boucle la plus interne, ou bien le case, dans lesquels elle se trouve :
  - On ne sort que d'un seul niveau
  - Ne concerne pas les if ... else
- Exemple :  

```
s = 0;
for(i=0;i<10;i++){    /*lecture d'au plus 10 entiers    */
    scanf("%d",&n);
    if(n<=0)           /*si entier non strictement positif */
        break;        /*on quitte la boucle          */
    s += n;
}
```

## continue

- Sert à sauter l'itération courante de la boucle la plus interne dans laquelle elle se trouve, et à passer à l'instruction suivante.

- Exemple :

```
s = 0;
for(i=0;i<10;i++){ /*lecture d'au plus 10 entiers */
scanf("%d",&n);
if(n<=0)           /*si entier non strictement positif */
    continue;      /*on passe au suivant */
s += n;
}
```

## Exercices

- Écrire trois programmes (max\_for.c, max\_while.c, max\_dowhile.c) qui calculent l'élément maximum du tableau d'entiers suivant :  

```
int tbl[10]={5,6,7,1,12,3,5,-2,10,3};
```

chaque programme utilisera respectivement une boucle for, une boucle while et une boucle do while.
- La fonction `getchar()` renvoie un caractère tapé au clavier. En utilisant cette fonction, écrire un programme qui indique si le caractère tapé est : une lettre minuscule, une lettre majuscule, ou un chiffre. S'il s'agit d'un autre caractère, le programme indique qu'il s'agit d'un autre caractère. Ce programme doit également s'arrêter dès que l'utilisateur tape le caractère %.
- Écrire, compiler et exécuter les programmes qui sont les solutions des exercices suivants :
  1. Convertir un entier en chaîne de caractères,
  2. Supprimer les commentaires d'un programme en langage C.