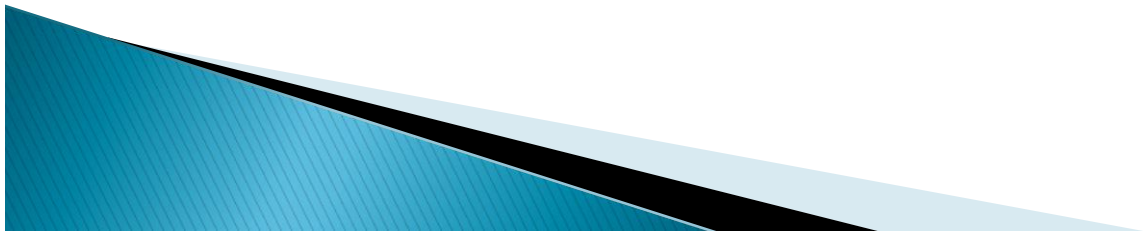
Les instructions simples

- Instruction vide :
  - **Syntaxe : ;**
- Instruction-expression :
  - **Syntaxe : <expression>;**
- Instruction-bloc :
  - **Syntaxe :**

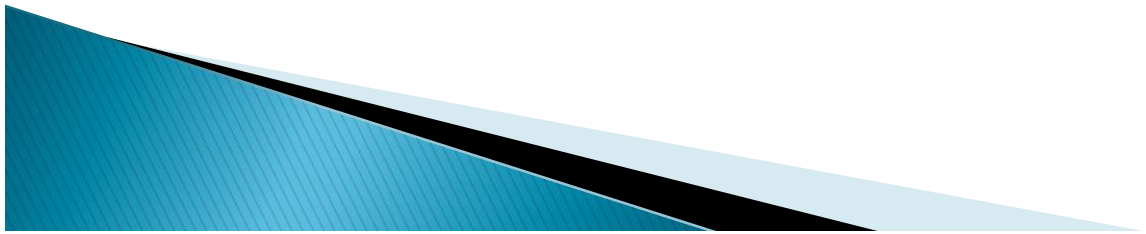
```
{
    déclarations
    instructions
}
```



Les Instructions Conditionnelles

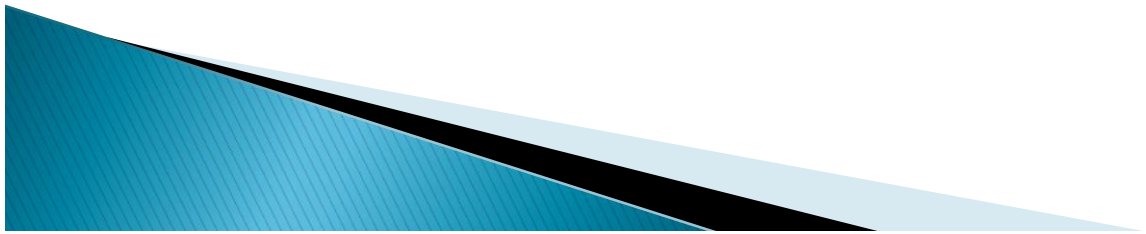
Instruction-if

- **Syntaxe :**
  - **if** (< expression >) < instruction1 > **else** < instruction2 >
  - **if** (< expression >) < instruction >



Les Instructions Conditionnelles

```
#include <stdio.h>
int main( )
   {
        int i, j ;
        scanf("%d %d", &i, &j) ;
        if(i < j) printf("%d est plus petit que %d \n", i, j);
        else
                {
                  if(i > j) printf("%d est plus grand que %d \n", i, j);
                  else printf("%d est egal a %d \n", i, j);
                }
        return 0;
   }
```

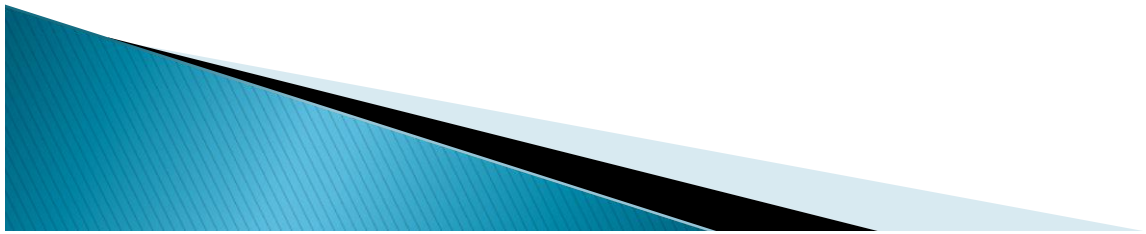


Les Instructions Conditionnelles

Instruction-switch

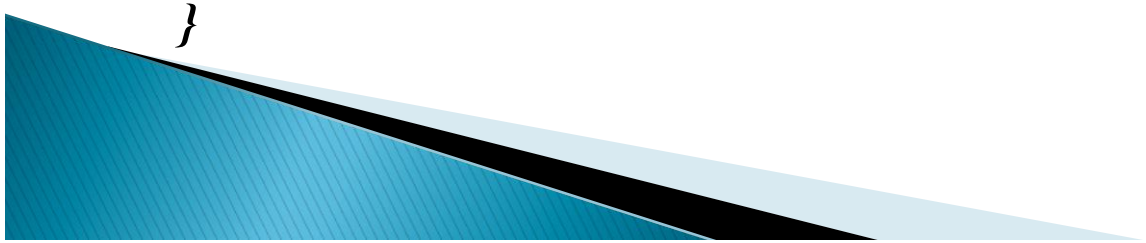
- **Syntaxe :**

```
switch(< expression >)
  {
      case < expression-constante1 > : < instruction1 >
      case < expression-constante2 > : < instruction2 >
      .
      .
      .
      case < expression-constantek > : < instructionk >
      default : < instruction >
  }
```



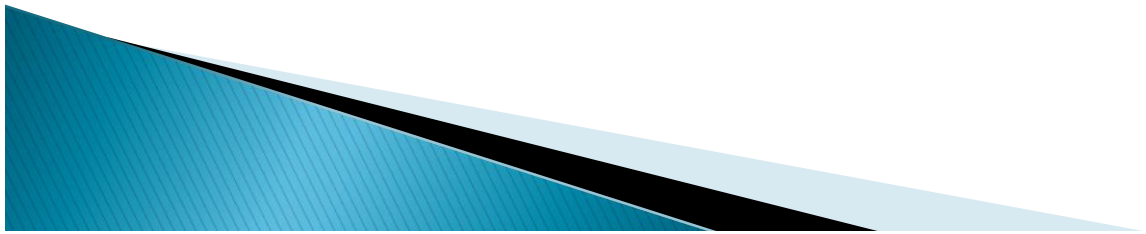
Les Instructions Conditionnelles

```
#include <stdio.h>
int main( )
   {
        int i;
        scanf("%d", &i) ;
        switch(i)
             {
                  case 1  : printf("je suis dans le cas 1\n");
                  case 2  : printf("je suis dans le cas 2\n");
                  case 3  : printf("je suis dans le cas 3\n");
                  default : printf("je ne suis ni dans le cas 1, ni dans le 2,
                                                              ni dans le 3\n");
             }
        return 0;
  }
```



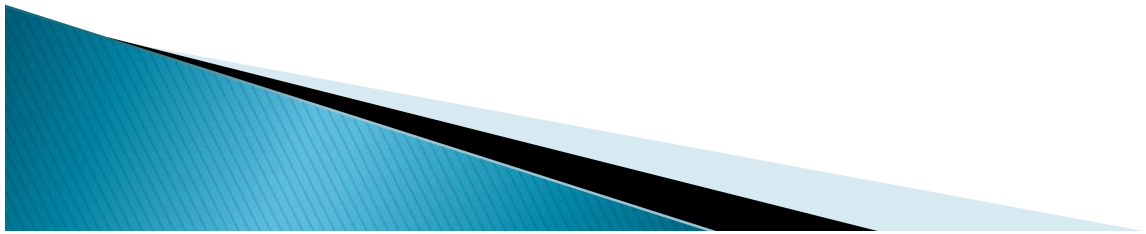
Les Instructions Conditionnelles

```
switch(< expression >)
  {
      case < expression-constante1 > : < instruction1 >
                                          break;
      case < expression-constante2 > : < instruction2 >
                                          break;
      .
      .
      .
      case < expression-constantek > : < instructionk >
                                          break;
      default : < instruction >
  }
```



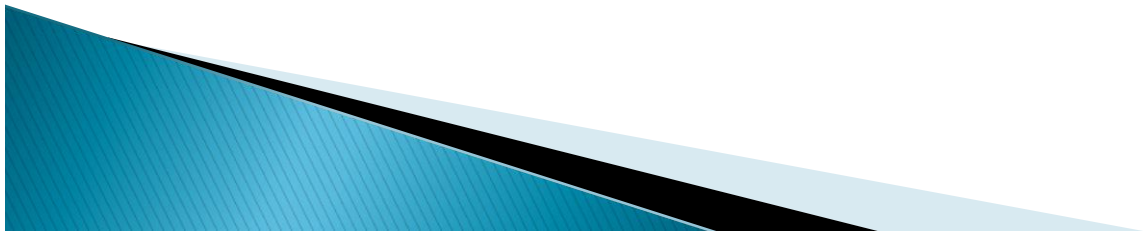
Les boucles

- **Instruction-while**
  - **Syntaxe : while**(< expression >) < instruction >
- **Instruction-dowhile**
  - **Syntaxe : do** < instruction > **while**(< expression >)**;**
- **Instruction-for**
  - **Syntaxe :**
    for(< expression1-opt >**;**< expression2-opt >**;**< expression3-opt >)
    < instruction >



Les boucles

```
#include <stdio.h>
int main( )
   {
          int i=10;
          while(i<=20)
          {
            printf("i = %d \n", i);
            i++;
          }
          return 0;
  }
```



Les boucles

- **Instruction break**
  - **Syntaxe : break;**
- **Instruction continue**
  - **Syntaxe : continue;**