

Introduction à la Programmation

Benoit Donnet
Année Académique 2023 - 2024



Agenda

- Introduction
- Chapitre 1: Bloc, Variable, Instruction Simple
- **Chapitre 2: Structures de Contrôle**
- Chapitre 3: Méthodologie de Développement
- Chapitre 4: Structures de Données
- Chapitre 5: Modularité du Code
- Chapitre 6: Pointeurs
- Chapitre 7: Allocation Dynamique

Agenda

- Chapitre 2: Structures de Contrôle
 - Condition
 - Itération

Agenda

- Chapitre 2: Structures de Contrôle
 - Condition
 - ✓ Principe
 - ✓ Condition Simple
 - ✓ Condition Complète
 - ✓ Condition Imbriquée
 - ✓ Condition Multiple
 - Itération

Principe

- Il est possible de tester la valeur d'une variable
 - ou de plusieurs variables
 - utilisation des opérateurs de comparaison
 - ✓ cfr. Chapitre 1
 - utilisation des opérateurs booléens
 - ✓ cfr. Chapitre 1
- En fonction du résultat, une instruction (ou plusieurs) peut être exécutée ou non
- Instruction **conditionnelle**

Condition Simple

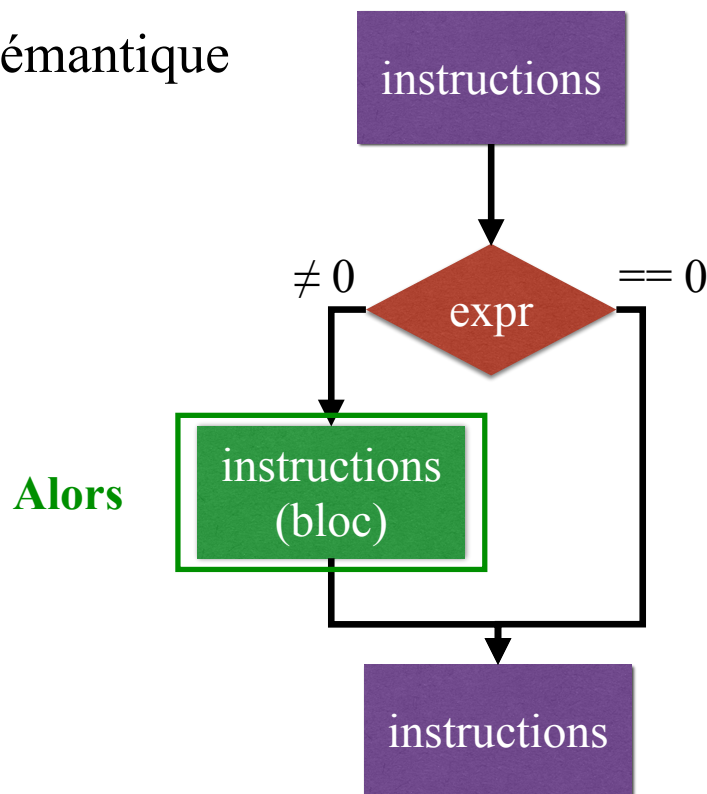
- L'instruction **if** est une instruction de contrôle qui permet d'orienter l'exécution du code en fonction d'une expression évaluée de manière booléenne
- Syntaxe

```
condition booléenne
mot-clé  if(expression) {    la condition est
    instructions;              entourée par des ()
}                                bloc d'instructions
```

- Effets
 - *expression* est évaluée
 - le bloc *instructions*; est exécuté uniquement si *expression* est évaluée à *vrai*

Condition Simple (2)

- Sémantique



Condition Simple (3)

- Exemple 1
 - comparaison de valeurs

```
#include <stdio.h>

int main(){
    int x, y;

    printf("Entrez des valeurs pour x et y: ");
    scanf("%d %d", &x, &y);

    if(x > 2*y){
        x = 2*y;
    }

    printf("%d\n", x);
} //fin programme
```

Condition Simple (4)

- Si le bloc d'instructions contient une seule instruction, les { } sont optionnelles

```
#include <stdio.h>

int main(){
    int x, y;

    printf("Entrez des valeurs pour x et y: ");
    scanf("%d %d", &x, &y);

    if(x > 2*y)
        x = 2*y;

    printf("%d\n", x);
} //fin programme
```

Condition Simple (5)

- Exemple 2
 - bloc "Alors" avec au moins 2 instructions

```
#include <stdio.h>

int main(){
    int x, y;

    printf("Entrez des valeurs pour x et y: ");
    scanf("%d %d", &x, &y);

    if(x > 2*y){
        x = 2*y;
        y = 0;
    }

    printf("%d\n", x);
} //fin programme
```

Condition Simple (6)

- On peut combiner plusieurs conditions à l'aide des opérateurs booléens
- Exemple

```
if((x > 0 && y>0) || (x < 0 && y < 0))  
    signe_xy = 1;
```

- Conseil
 - utilisez des parenthèses pour ne pas vous tromper dans les priorités

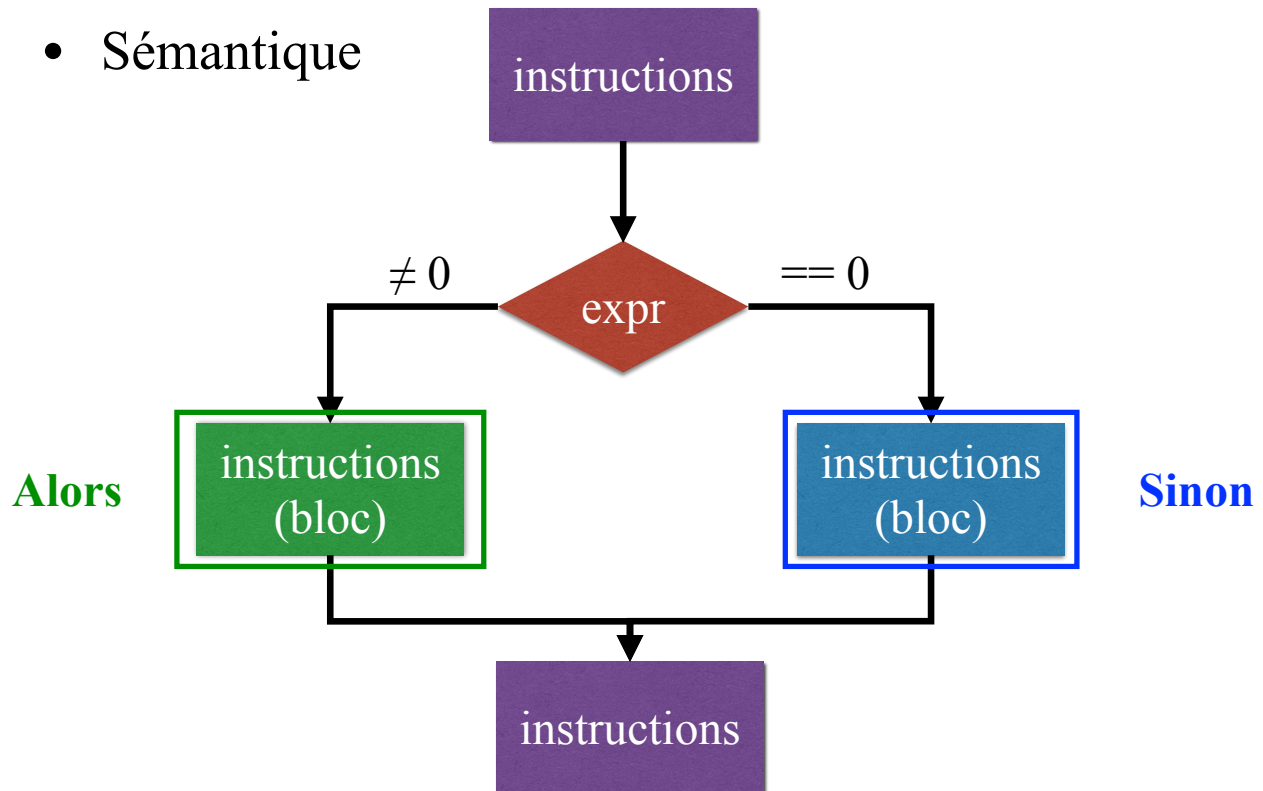
Condition Complète

- Supposons que l'on ait une condition x
 - Si x est vérifiée, Alors on exécute une action a
 - Sinon, on exécute une autre action b
- Instruction **conditionnelle complète**
- Syntaxe

```
      mots-clés      if(expression){  
                      instructions1;  
      bloc d'instructions correspondant au Sinon }else{  
                      instructions2;  
                      }  
      bloc d'instructions correspondant au Alors
```

Condition Complète (2)

- Sémantique



Condition Complète (3)

- Exemple

```
#include <stdio.h>

int main(){
    unsigned short age;
    printf("Entrez votre âge: ");
    scanf("%hu", &age);

    if(age >= 18)
        printf("Vous êtes majeur!\n");
    else
        printf("Vous êtes mineur!\n");
} //fin programme
```

Condition Imbriquée


- Le bloc d'instructions de chaque clause d'une condition (**Alors** et **Sinon**) peut contenir n'importe quel type d'instruction
 - même une autre condition
 - **condition imbriquée**
- Exemple

```
if (age >= 99)
    printf("Vous êtes trop âgé!\n");
else{
    if (age < 18)
        printf("Interdit aux mineurs!\n");
    else
        printf("Vous pouvez entrer!\n");
}
```

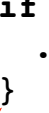
Condition Imbriquée (2)

- Une clause **else** se rapporte à l'instruction **if** la plus proche appartenant au même bloc
- Exemple

```
if (expr1)
    if (expr2){
        ...
    }
else{
    ...
}
```



```
if (expr1){
    if (expr2){
        ...
    }
}
else{
    ...
}
```



Condition Imbriquée (3)

```
#include <stdio.h>

int main(){
    float a, b, c;
    printf("Entrez 3 nombres réels: ");
    scanf("%f %f %f", &a, &b, &c);

    if(a > b){
        if(a > c)
            printf("Le maximum est: %f\n", a);
        else
            printf("Le maximum est: %f\n", c);
    }else{
        if(b > c)
            printf("Le maximum est: %f\n", b);
        else
            printf("Le maximum est: %f\n", c);
    }
} //fin programme
```

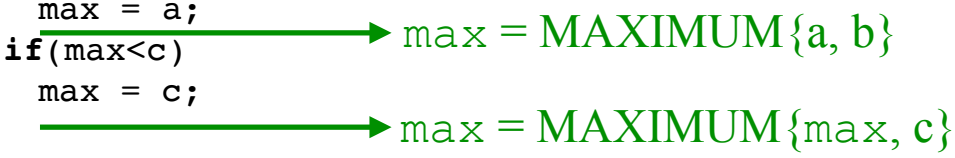
Condition Imbriquée (4)

- Version plus esthétique

```
#include <stdio.h>

int main(){
    float a, b, c, max;
    printf("Entrez 3 nombres réels: ");
    scanf("%f %f %f", &a, &b, &c);

    if(a < b)
        max = b;
    else
        max = a;
    if(max < c)
        max = c;
    printf("Le maximum est: %f\n", max);
} //fin programme
```



Condition Multiple

```
if(age == 2){
    printf("Salut bébé!\n");
}else{
    if(age == 6){
        printf("Salut gamin!\n");
    }else{
        if(age == 16){
            printf("Salut ado!\n");
        }else{
            if(age == 18){
                printf("Salut adulte!\n");
            }else{
                if(age == 68)
                    printf("Salut vieux!\n");
                else
                    printf("Je n'ai pas de phrase pour ton âge!\n");
            }
        }
    }
}
```

Condition Multiple (2)

- Comment faire efficacement plusieurs tests d'égalité sur une même variable et exécuter le bon bloc de code?
 - instructions conditionnelles imbriquées sont difficiles à lire
 - ✓ `if(cond){instr} else if(cond){instr}...`
- Instruction **switch**

Condition Multiple (3)

- Syntaxe

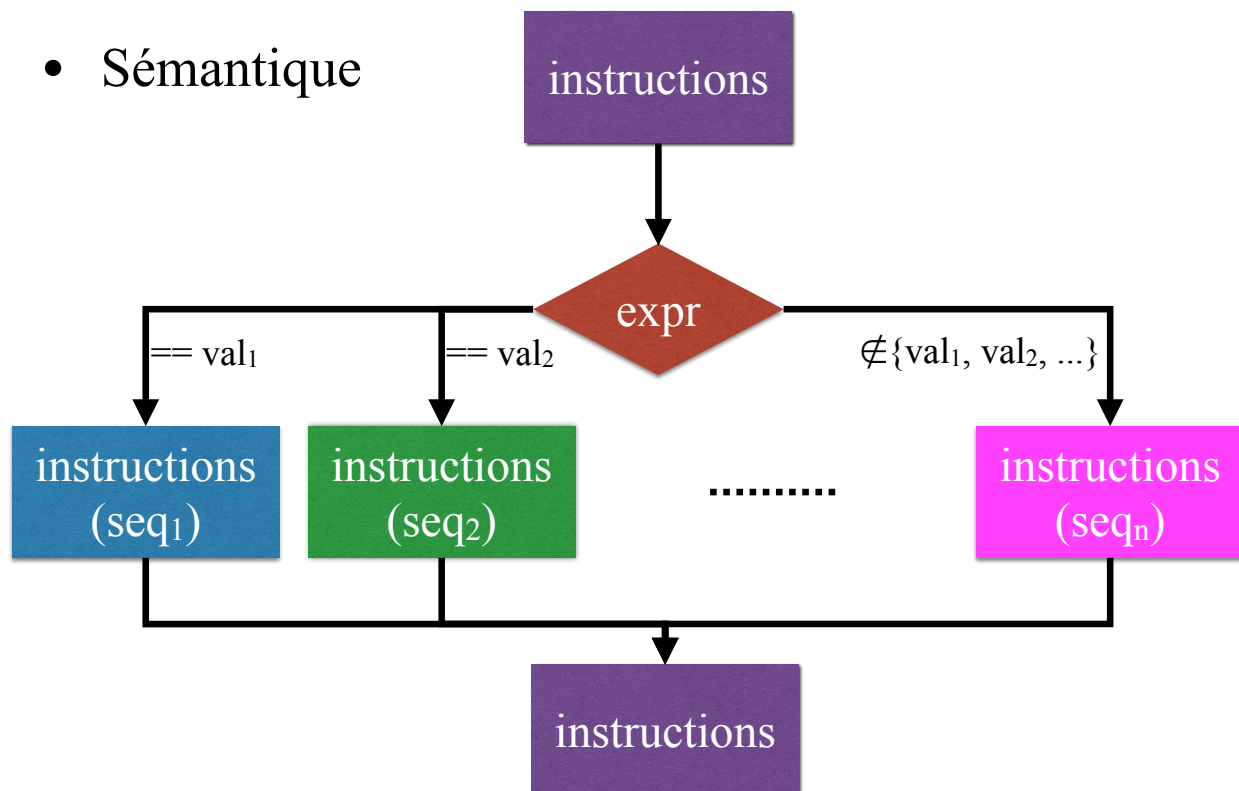
```
switch(expr) {  
    case val1:  
        seq1;  
        break;  
    case val2:  
        seq2;  
        break;  
    ...  
    [ default:  
        seqn; ]  
}
```

expr doit être une variable de type int ou un char
seq₁ à exécuter si expr vaut val₁
seq₂ à exécuter si expr vaut val₂
seq_n à exécuter sinon

mots-clés

Condition Multiple (4)

- Sémantique



Condition Multiple (5)

```
switch(age){  
  case 2:  
    printf("Salut bébé!\n");  
    break;  
  case 6:  
    printf("Salut gamin!");  
    break;  
  case 16:  
    printf("Salut ado!\n");  
    break;  
  case 18:  
    printf("Salut adulte!\n");  
    break;  
  case 68:  
    printf("Salut vieux!\n");  
    break;  
  default:  
    printf("Je n'ai pas de phrase pour ton âge!\n");  
} //fin switch
```

Condition Multiple (6)

- Exemple complet

```
#include <stdio.h>  
  
int main(){  
  char choix;  
  printf("Menu: faites un choix:\n");  
  printf("\tAfficher la liste des clients: a\n");  
  printf("\tAfficher les données d'un client: b\n");  
  printf("\tCréer un client: c\n");  
  
  scanf("%c", &choix);  
  
  //à suivre  
} //fin main()
```

Condition Multiple (7)

```
switch(choix){  
  case 'a':  
    printf("Affichage de la liste\n");  
    //ajouter le code d'affichage ici  
    break;  
  case 'b':  
    printf("Veuillez entrer le nom du client\n");  
    //ajouter ici le code  
    break;  
  case 'c':  
    printf("Veuillez entrer les données\n");  
    //ajouter ici le code  
    break;  
  default:  
    printf("erreur\n");  
} //fin switch  
} //fin programme
```

Exercices

- Ecrire un programme qui lit deux entiers au clavier et les affiche dans l'ordre croissant
- Ecrire un programme qui lit deux entiers, a et b, et donne le choix à l'utilisateur
 1. savoir si la somme $a+b$ est paire
 2. savoir si le produit $a \times b$ est pair
 3. connaître le signe de la somme $a+b$
 4. connaître le signe du produit $a \times b$

Agenda

- Chapitre 2: Structures de Contrôle
 - Condition
 - Itération
 - ✓ Principe
 - ✓ while
 - ✓ for
 - ✓ do ... while

Principe

- Il est impératif de pouvoir exécuter plusieurs fois la même séquence d'instructions
- Exemple
 - vous êtes mélomane
 - vous voulez afficher les paroles du tube interplanétaire *Highway to Hell*



© B. Donnet -- SdF 2009

Principe (2)

- Code (fin couplet 1 + refrain + début couplet 2)

```
#include <stdio.h>
```

```
int main(){
```

```
    printf("Going down, party time\n");
```

```
    printf("My friends are gonna be there too\n");
```

```
    printf("Highway to Hell!\n");
```

```
    printf("Highway to Hell!\n");
```

```
    printf("Highway to Hell!\n");
```

```
    printf("Highway to Hell!\n");
```

```
    printf("No stop signs, speed limit\n");
```

```
} //fin programme
```

pas pratique de réécrire
plusieurs fois la même
ligne

Principe (3)

- En programmation, il existe une structure de contrôle particulière permettant de répéter plusieurs fois un même bloc d'instructions
 - **boucle**
 - **itération**
- On va aborder 3 types de boucle
 - while
 - ✓ tant que
 - for
 - ✓ pour
 - do ... while
 - ✓ faire ... tant que

while

- Instruction **while**
 - répéter une instruction (ou un bloc d'instructions)
 - tant qu'une condition (expression booléenne) est satisfaite
 - condition évaluée avant chaque itération

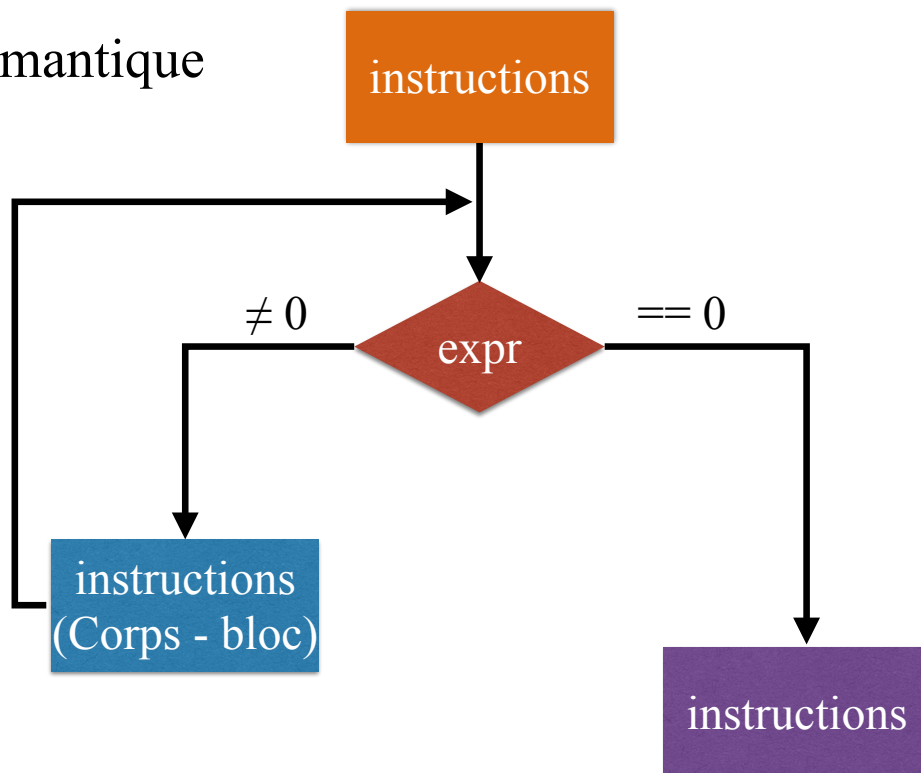
```
Tant Que(condition_x)  Gardien de Boucle  
    action_a
```

- Syntaxe

```
                                expression booléenne  
mot-clé  while(expression){    la condition est  
                                instructions;    entourée par des ()  
                                }    corps de la boucle (bloc)
```

while (2)

- Sémantique



while (3)

- Vocabulaire

- **Gardien de Boucle**

- ✓ expression booléenne évaluée à chaque tentative d'entrée dans la boucle
- ✓ doit être vrai pour autoriser l'entrée dans la boucle

- **Critère d'Arrêt**

- ✓ négation du Gardien de Boucle
- ✓ permet de sortir de la boucle

while (4)

- Retour sur le Highway to Hell

- Comment écrire une boucle?

- 4 étapes

1. déclarer une variable qui va servir de compteur et l'initialiser
 - `int i = 0;`
2. déterminer le nombre de tours de la boucle
 - 4 dans le cas de Highway to Hell
3. déterminer le Gardien de Boucle
 - `i < 4`
4. déterminer le Corps de Boucle
 - `printf();`
 - incrémentation du compteur

while (5)

- Code (fin couplet 1 + refrain + début couplet 2)

```
#include <stdio.h>
```

```
int main(){
```

```
    int i = 0;  déclaration et initialisation du compteur
```

```
    printf("Going down, party time\n");
```

```
    printf("My friends are gonna be there too\n");
```

```
    while(i<4){  gardien de la boucle
```

```
        printf("Highway to Hell\n");
```

```
        i++;
```

Corps de la boucle

```
    } //fin while - i
```

incrémentation du compteur

```
    printf("No stop signs, speed limit\n");
```

```
    } //fin programme
```

while (5)

i :



```
#include <stdio.h>
```

```
int main(){
```

```
    int i = 0;
```

```
    printf("Going down, party time\n");
```

```
    printf("My friends are gonna be there  
too\n");
```

```
    while(i<4){
```

```
        printf("Highway to Hell\n");
```

```
        i++;
```

```
    } //fin while - i
```

```
    printf("No stop signs, speed limit\n");
```

```
    } //fin programme
```

```
$> ./highway
```

while (5)

i: 0

```
#include <stdio.h>

int main(){
    int i = 0;

    printf("Going down, party time\n");
    printf("My friends are gonna be there
too\n");

    while(i<4){
        printf("Highway to Hell\n");
        i++;
    }//fin while - i

    printf("No stop signs, speed limit\n");
}//fin programme
```

\$>./highway

while (5)

i: 0

```
#include <stdio.h>

int main(){
    int i = 0;

    printf("Going down, party time\n");
    printf("My friends are gonna be there
too\n");

    while(i<4){
        printf("Highway to Hell\n");
        i++;
    }//fin while - i

    printf("No stop signs, speed limit\n");
}//fin programme
```

\$>./highway
Going down, party time

while (5)

i: 0

```
#include <stdio.h>

int main(){
    int i = 0;

    printf("Going down, party time\n");
    printf("My friends are gonna be there
too\n");

    while(i<4){
        printf("Highway to Hell\n");
        i++;
    }//fin while - i

    printf("No stop signs, speed limit\n");
}//fin programme
```

```
$>./highway
Going down, party time
My friends are gonna
be there too
```

while (5)

i: 0

```
#include <stdio.h>

int main(){
    int i = 0;

    printf("Going down, party time\n");
    printf("My friends are gonna be there
too\n");

    while(i<4){
        printf("Highway to Hell\n");
        i++;
    }//fin while - i

    printf("No stop signs, speed limit\n");
}//fin programme
```

```
$>./highway
Going down, party time
My friends are gonna
be there too
```

while (5)

i: 0

```
#include <stdio.h>

int main(){
    int i = 0;

    printf("Going down, party time\n");
    printf("My friends are gonna be there
too\n");

    while(i<4){
        printf("Highway to Hell\n");
        i++;
    }//fin while - i

    printf("No stop signs, speed limit\n");
}//fin programme
```

```
$>./highway
Going down, party time
My friends are gonna
be there too
Highway to Hell
```

while (5)

i: 1

```
#include <stdio.h>

int main(){
    int i = 0;

    printf("Going down, party time\n");
    printf("My friends are gonna be there
too\n");

    while(i<4){
        printf("Highway to Hell\n");
        i++;
    }//fin while - i

    printf("No stop signs, speed limit\n");
}//fin programme
```

```
$>./highway
Going down, party time
My friends are gonna
be there too
Highway to Hell
```

while (5)

i: 1

```
#include <stdio.h>

int main(){
    int i = 0;

    printf("Going down, party time\n");
    printf("My friends are gonna be there
too\n");

    while(i<4){
        printf("Highway to Hell\n");
        i++;
    }//fin while - i

    printf("No stop signs, speed limit\n");
}//fin programme
```

```
$>./highway
Going down, party time
My friends are gonna
be there too
Highway to Hell
```

while (5)

i: 1

```
#include <stdio.h>

int main(){
    int i = 0;

    printf("Going down, party time\n");
    printf("My friends are gonna be there
too\n");

    while(i<4){
        printf("Highway to Hell\n");
        i++;
    }//fin while - i

    printf("No stop signs, speed limit\n");
}//fin programme
```

```
$>./highway
Going down, party time
My friends are gonna
be there too
Highway to Hell
Highway to Hell
```

while (5)

i: 2

```
#include <stdio.h>

int main(){
    int i = 0;

    printf("Going down, party time\n");
    printf("My friends are gonna be there
too\n");

    while(i<4){
        printf("Highway to Hell\n");
        i++;
    }//fin while - i

    printf("No stop signs, speed limit\n");
}//fin programme
```

```
$>./highway
Going down, party time
My friends are gonna
be there too
Highway to Hell
Highway to Hell
```

while (5)

i: 2

```
#include <stdio.h>

int main(){
    int i = 0;

    printf("Going down, party time\n");
    printf("My friends are gonna be there
too\n");

    while(i<4){
        printf("Highway to Hell\n");
        i++;
    }//fin while - i

    printf("No stop signs, speed limit\n");
}//fin programme
```

```
$>./highway
Going down, party time
My friends are gonna
be there too
Highway to Hell
Highway to Hell
```

while (5)

i: 2

```
#include <stdio.h>

int main(){
    int i = 0;

    printf("Going down, party time\n");
    printf("My friends are gonna be there
too\n");

    while(i<4){
        printf("Highway to Hell\n");
        i++;
    }//fin while - i

    printf("No stop signs, speed limit\n");
} //fin programme
```

```
$>./highway
Going down, party time
My friends are gonna
be there too
Highway to Hell
Highway to Hell
Highway to Hell
```

while (5)

i: 3

```
#include <stdio.h>

int main(){
    int i = 0;

    printf("Going down, party time\n");
    printf("My friends are gonna be there
too\n");

    while(i<4){
        printf("Highway to Hell\n");
        i++;
    }//fin while - i

    printf("No stop signs, speed limit\n");
} //fin programme
```

```
$>./highway
Going down, party time
My friends are gonna
be there too
Highway to Hell
Highway to Hell
Highway to Hell
```


while (5)

i: 3

```
#include <stdio.h>

int main(){
    int i = 0;

    printf("Going down, party time\n");
    printf("My friends are gonna be there
too\n");

    while(i<4){
        printf("Highway to Hell\n");
        i++;
    }//fin while - i

    printf("No stop signs, speed limit\n");
}//fin programme
```

```
$>./highway
Going down, party time
My friends are gonna
be there too
Highway to Hell
Highway to Hell
Highway to Hell
```

while (5)

i: 3

```
#include <stdio.h>

int main(){
    int i = 0;

    printf("Going down, party time\n");
    printf("My friends are gonna be there
too\n");

    while(i<4){
        printf("Highway to Hell\n");
        i++;
    }//fin while - i

    printf("No stop signs, speed limit\n");
}//fin programme
```

```
$>./highway
Going down, party time
My friends are gonna
be there too
Highway to Hell
Highway to Hell
Highway to Hell
Highway to Hell
```

while (5)

i:

4

```
#include <stdio.h>

int main(){
    int i = 0;

    printf("Going down, party time\n");
    printf("My friends are gonna be there
too\n");

    while(i<4){
        printf("Highway to Hell\n");
        i++;
    }//fin while - i

    printf("No stop signs, speed limit\n");
}//fin programme
```

```
$>./highway
Going down, party time
My friends are gonna
be there too
Highway to Hell
Highway to Hell
Highway to Hell
Highway to Hell
```

while (5)

i:

4

```
#include <stdio.h>

int main(){
    int i = 0;

    printf("Going down, party time\n");
    printf("My friends are gonna be there
too\n");

    while(i<4){
        printf("Highway to Hell\n");
        i++;
    }//fin while - i

    printf("No stop signs, speed limit\n");
}//fin programme
```

```
$>./highway
Going down, party time
My friends are gonna
be there too
Highway to Hell
Highway to Hell
Highway to Hell
Highway to Hell
```

while (5)

i: 4

```
#include <stdio.h>

int main(){
    int i = 0;

    printf("Going down, party time\n");
    printf("My friends are gonna be there too\n");

    while(i<4){
        printf("Highway to Hell\n");
        i++;
    }//fin while - i

    printf("No stop signs, speed limit\n");
}//fin programme
```

```
$>./highway
Going down, party time
My friends are gonna
be there too
Highway to Hell
Highway to Hell
Highway to Hell
Highway to Hell
No stop signs, speed
limit
```

while (6)

- Exemple
 - calcul de n^k , avec (par exemple) $n=2$ et $k = 100$

```
#include <stdio.h>

int main(){
    unsigned int n = 2, k = 1, exp = 1;

    while(k <= 100){
        exp *= n; // équivalent à exp = exp * n;
        k++;
    }//fin while - k

    printf("%u\n", exp);
}//fin programme
```

while (7)

- Exécution de la boucle `while` ($n = 2$)

k	<code>exp = exp × n</code>
1	<code>2 = 1 × 2</code>
2	<code>4 = 2 × 2</code>
3	<code>8 = 4 × 2</code>
4	<code>16 = 8 × 2</code>
5	<code>32 = 16 × 2</code>
...	...

- Ce code fournit-il un résultat correct?

for

- Instruction **for**
 - Permet de spécifier explicitement les opérations à effectuer
 - ✓ avant d'entrer dans la boucle (`instr1`)
 - initialisation du compteur
 - ✓ afin de décider s'il faut continuer (ou non) à itérer (`expr`)
 - Gardien de Boucle
 - ✓ entre deux itérations (`instr2`)
 - incrémentement du compteur
 - ✓ dans le Corps de Boucle (`instr3`)
- Syntaxe

```
mot-clé for(instr1; expr; instr2){  
    instr3;           instructions simples  
}  Corps de Boucle
```

Gardien de Boucle

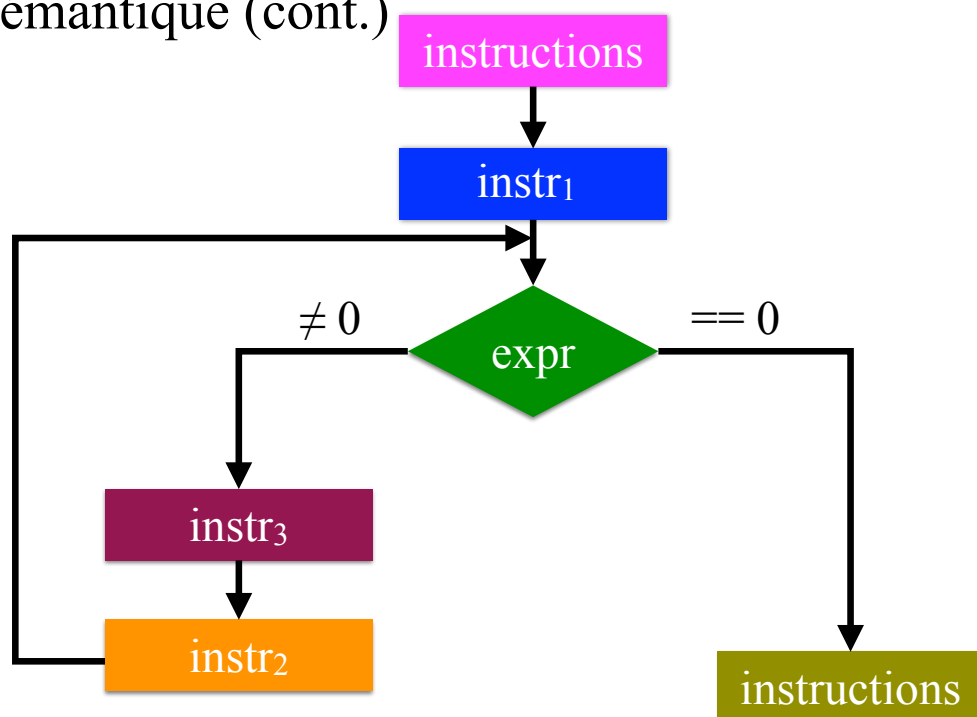
for (2)

- Sémantique

<pre>for(instr₁; expr; instr₂) instr₃;</pre>	<pre>instr₁; while(expr) { instr₃; instr₂; }</pre>
---	---

for (3)

- Sémantique (cont.)



for (4)

- *Highway to Hell*

```
#include <stdio.h>

int main(){
    int i;

    printf("Going down, party time\n");
    printf("My friends are gonna be there too\n");

    for(i=0; i<4; i++)
        printf("Highway to Hell\n");

    printf("No stop signs, speed limit\n");
} //fin programme
```

for (5)

- n^k

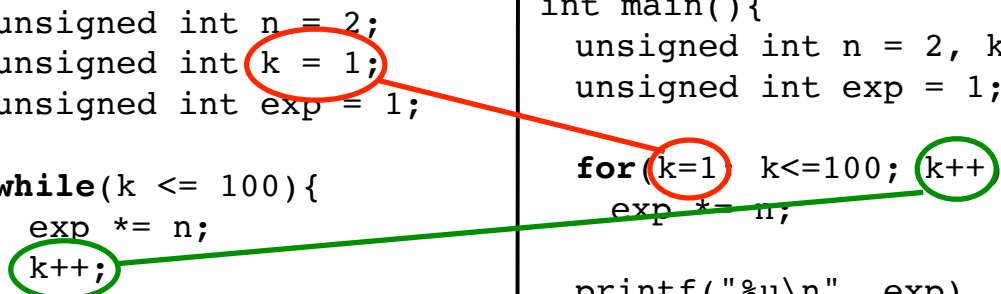
```
#include <stdio.h>
int main(){
    unsigned int n = 2;
    unsigned int k = 1;
    unsigned int exp = 1;

    while(k <= 100){
        exp *= n;
        k++;
    } //fin while - k
    printf("%u\n", exp);
} //fin programme
```

```
#include <stdio.h>
int main(){
    unsigned int n = 2, k;
    unsigned int exp = 1;

    for(k=1; k<=100; k++)
        exp *= n;

    printf("%u\n", exp)
} //fin programme
```



for (6)

- Les instructions $instr_1, instr_2, instr_3$ sont optionnelles
- $expr$ peut également être omise
 - boucle infinie
- `for (; ;)`

do ... while

- Instruction **do ... while**
 - évalue la condition permettant de poursuivre l'itération après celle-ci

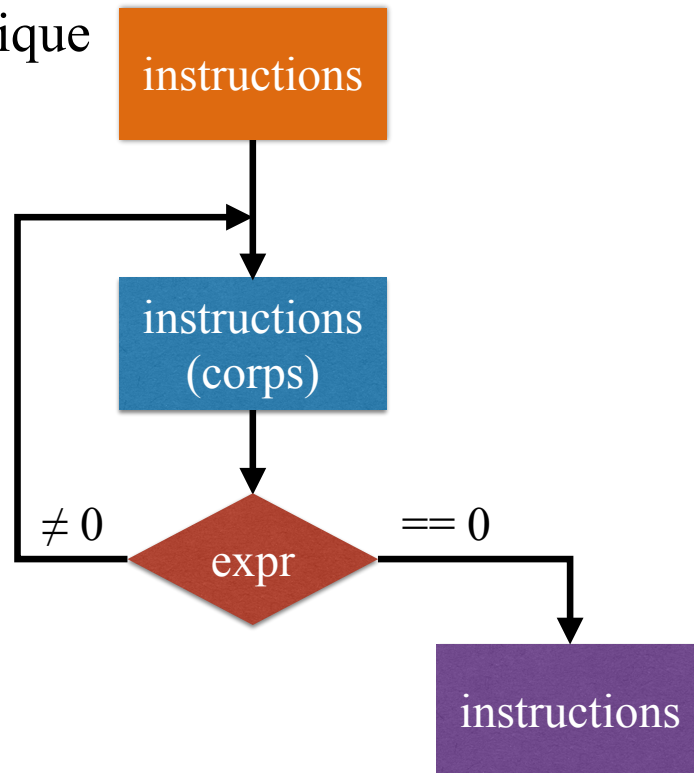
```
Faire  
  action_a  
Tant Que (condition_x)
```

- Syntaxe

```
mots-clés      do{          Corps de Boucle  
                instr;  
            }while(expr);  Gardien de Boucle
```

do ... while (2)

- Sémantique



Exercices

- Ecrire un programme qui lit au clavier un entier n et calcule sa factorielle ($n! = n \times (n-1)!, n > 1$)
- Ecrire un programme qui calcule
 - $s = 1 + 2^3 + 3^3 + \dots + n^3$
 - avec n ayant été introduit au clavier
- Ecrire un programme qui affiche la valeur en degrés Celsius correspondant aux valeurs 0, 10, 20, 30, ..., 300 degrés Fahrenheit.
 - $C \sim 0.55556 \times (F - 32)$
- Ecrire un programme qui calcule le $n^{\text{ième}}$ terme de la suite de Fibonacci
 - $u_0 = 0, u_1 = 1, u_2 = 1$
 - $u_n = u_{n-1} + u_{n-2}$, avec $n \geq 2$