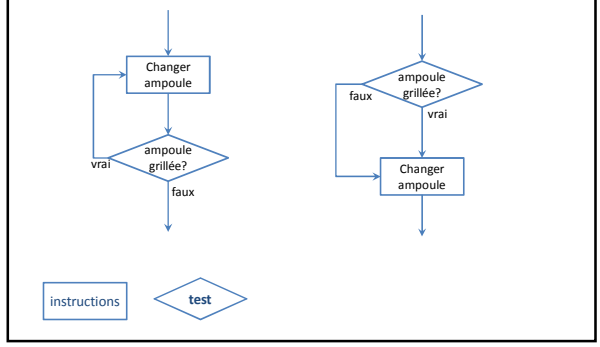
Informatique S1
Programmation C

- *Objectifs de la séance*
  - **Présentation des boucles (Partie I)**
- *Concepts*
  - Opérateurs relationnels >,<,>=,<=, !=, ==
  - Boucle while
  - Boucle do-while
- *Concepts complémentaires*
  - Notion de typecast
  - Indentation

Boucles

- Boucles
  - Répéter un bloc d'instructions tant qu'une condition logique soit vrai
  - On ne sait pas à priori combien de fois le bloc doit être exécuté
- Exemples d'usage :
  - Calculer x^Y avec X et Y fournis par l'utilisateur
  - Calculer le factoriel de n (1 x 2 x … x n)
  - Calculer la moyenne d'un ensemble
  - …

Boucles



Opérateurs relationnels

- Pour comparer deux valeurs

Vrai ou faux
Pas de type booléen en C
Faux → 0 (zero)
Vrai → ≠ 0

==	égal à	`a == b`
!=	différent de	`a != b`
<	inférieur à	`a < b`
<=	inférieur ou égal à	`a <= b`
>	supérieur à	`a > b`
>=	supérieur ou égal à	`a >= b`

Boucle while

- Format :

  while (*test*)

  {

  bloc d'instructions ;

  }

Boucle infinie

Le test doit être vrai à un moment donné ou on restera bloqué !!

```
while (i < y) {
      expo = expo * x;
      i = i + 1;
}
```

Boucle while

```
1  #include <stdio.h>
2
3  int main (int argc, char argv[]) {
4      int x, y;
5      float expo;
6      int i;
7
8      /* point d'observation 1 */
9      printf ("Entrez x : ");
10     scanf ("%d", &x);
11     printf ("Entrez y : ");
12     scanf ("%d", &y);
13
14     expo = 1;
15     i = 0;
16
17     /* point d'observation 2 */
18
19     while (i < y) {
20         expo = expo * x;
21         i = i + 1;
22     }
23
24     /* point d'observation 4 */
25
26     printf ("x ^ y = %f \n", expo);
27 }
```

Test : i < y

instructions

/*point d'observation 3 */

Trace d'exécution				
	expo	i	y	x
PO 2	1	0	2	2
PO 3	2	1	2	2
PO 3	4	2	2	2
PO 4	4	2	2	2

Boucle do-while

- Format :

  do

  {

  bloc d'instructions ;

  } while (*test*) ;

Boucle infinie

Le test doit être vrai à un moment donné ou on restera bloqué !!

```
do {
      expo = expo * x;
      i = i + 1;
} while (i < y);
```

Boucle do-while

```
1  #include <stdio.h>
2
3  int main (int argc, char argv[]) {
4      int x, y;
5      float expo;
6      int i;
7
8      /* point d'observation 1 */
9      printf ("Entrez x : ");
10     scanf ("%d", &x);
11     printf ("Entrez y : ");
12     scanf ("%d", &y);
13
14     expo = 1;
15     i = 0;
16
17     /* point d'observation 2 */
18
19     do {
20         expo = expo * x;
21         i = i + 1;
22     } while (i < y);
23
24     /* point d'observation 4 */
25
26     printf ("x ^ y = %f \n", expo);
27 }
```

instructions

/*point d'observation 3 */

Test : i < y

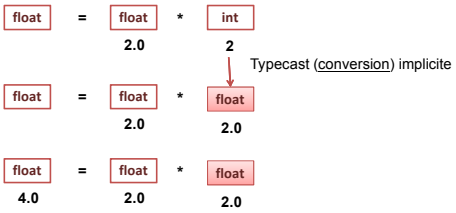
Trace d'exécution				
	expo	i	y	x
PO 2	1	0	2	2
PO 3	2	1	2	2
PO 3	4	2	2	2
PO 4	4	2	2	2

Typecast

- Conversion de type
  - int → float 3 → 3.0
  - float → int 3.5 → 3
- Implicite
  - Automatique
  - Pas assez fiable
- **Explicite**
  - On indique explicitement le nouveau type
  - **(newtype)**

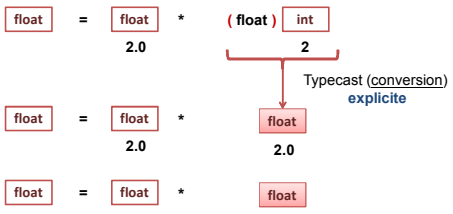
Typecast implicite

```
int x, y;
float expo;
...
    expo = expo * x ;
...
```



Typecast explicite

```
int x, y;
float expo;
...
    expo = expo * (float) x ;
...
```



Typecast explicite

- Avantage :
  - Visibilité

```
int  i = 4;
float x = 3.45;


i = x;
```

```
int  i = 4;
float x = 3.45;


i = (float) x;
```