HEH.be Sciences et technologies

Base de programmation

- BA1 Informatique
Johan Depréter – johan.depreter@heh.be

HEH.be Sciences et technologies

- Inscription eCampus

Python4Ever

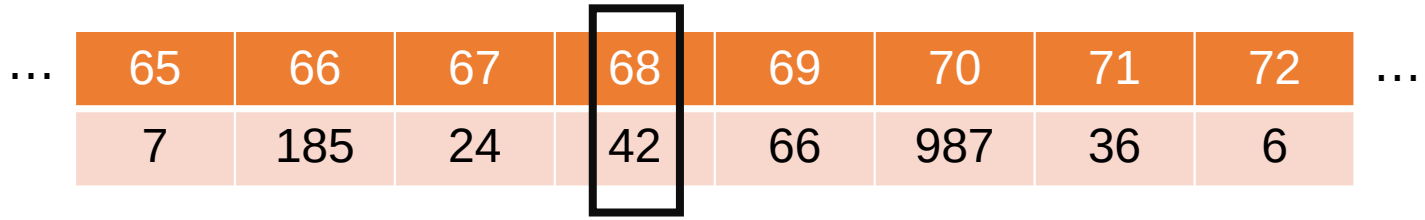
HEH.be Sciences et technologies

Chapitre 2

Les principes de base

Les variables

- Principe :



Nom de la variable : number

Valeur : 42

Les variables

- Une variable est un nom qui permet d'accéder à un emplacement dans la mémoire centrale.
- Les conventions dépendent du langage de programmation.
- De manière générale :
  - Lettres
  - Pas d'espace
  - Chiffres sauf pour débuter

Exemples :
a, tab, var1,
randNumb, …

Le type d'une variable

- Savoir comment la valeur a été codée.
- La place en mémoire va dépendre du type.
- Types fréquents :
  - Entiers
  - Réels
  - Caractères
  - Booléens

Le type d'une variable

- Typage statique / dynamique

- Typage explicite / implicite

```
int i = 0;
```

```
i = 0
```

- Typage fort / faible

Les variables

- La déclaration

  Dans le cas où le langage est explicite, il faut prévoir la place en mémoire.

- L'affectation

  Instruction qui consiste à placer une valeur dans une variable.

  Exemples :

  number = 127;

L’affectation

- $a = b \neq b = a$
- $a = a + 1$
- ~~$a + 5 = 3$~~

L’initialisation

- $a = a + 1$

  Combien vaut *a* ?

- Important d’initialiser pour ne pas obtenir des résultats incohérents

Exercices

- Exercice n°1 :

n1 = 5
n2 = 7
n1 = *n2*
n2 = *n1*

Que vaut *n1* et *n2* ?
n1 = 7 et *n2* = 7

Exercices

- Exercice n°2 :

$$a = 5$$
$$b = a + 7$$
$$a = a - 1$$
$$b = a + 3$$

Que vaut a et b ?

$a = 4$ et $b = 7$

Les opérations

- Opérations fréquentes :
  - +
  - -
  - *
  - /
  - %

- Règles de priorités des opérations

Opérateurs

- Opérateurs de comparaison :

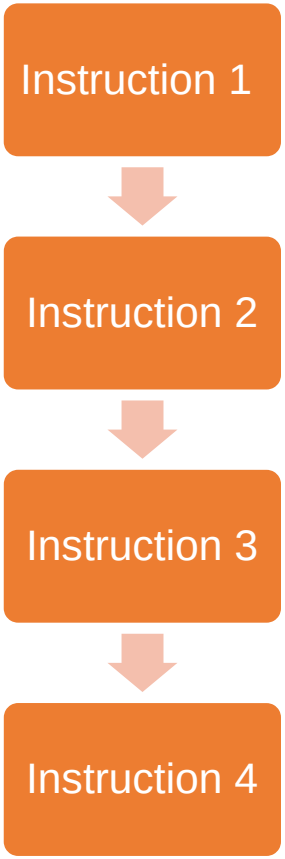
Opération	Opérateur
Est plus grand que	>
Est plus petit que	<
Est plus grand ou égal à	>=
Est plus petit ou égal à	<=
Est égale à	==
Est différent de	!=

Opérateurs

- Opérateurs logiques :

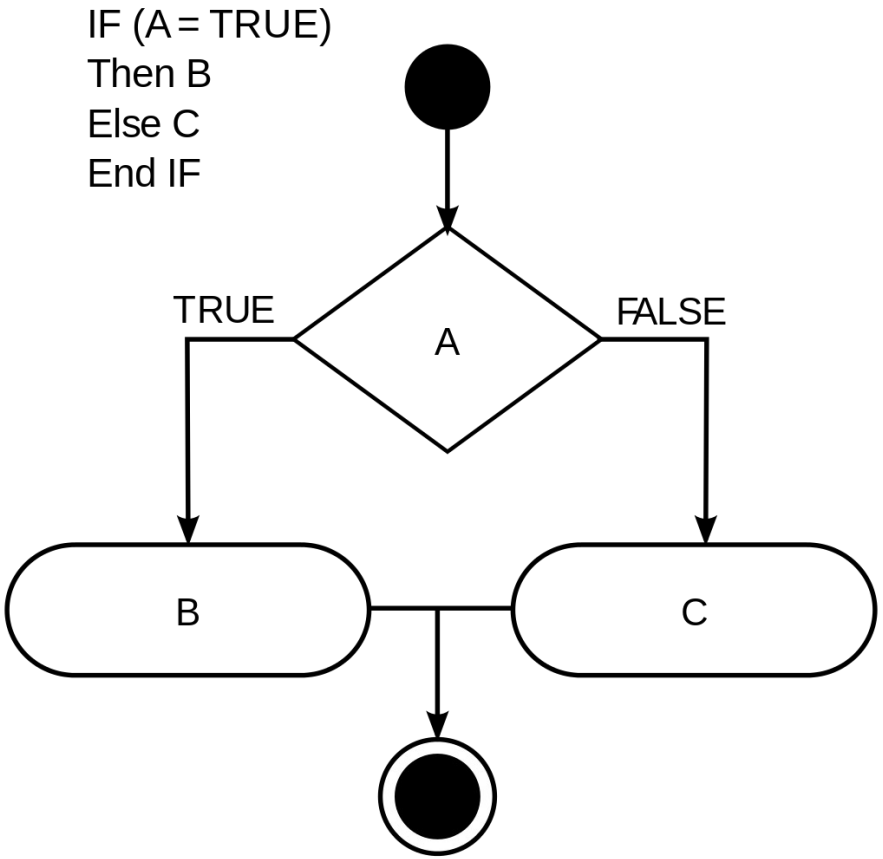
Opération	Opérateur
ET	and / &&
OU	or / ||
NON	not / !

Les structures conditionnelles



Les structures conditionnelles

- Permet de choisir un bloc d'instructions en fonction d'une condition ou d'une valeur.



Les structures conditionnelles

En C :

```
if (test expression)
{
        // run code if test expression is
true
}
else
{
        // run code if test expression is
false }
```

En Python :

```
if test expression:
    // Body of if
else:
    // Body of else
```

Les structures conditionnelles

- Possibilité de multiplier les conditions à l'aide des opérateurs logique
- *if (a>=0) and (a<10)* :
- If… else if / elif … else

Les structures conditionnelles

- Switch

```
switch(x)
{
  case 1:
          printf(" nous sommes dans le cas où x = 1");
          break;
  case 2:
          printf(" nous sommes dans le cas où x = 2");
          break;
  case 3:
          printf(" nous sommes dans le cas où x = 3");
          break;
  case 4:
          printf(" nous sommes dans le cas où x = 4");
          break;
}
```

Les structures conditionnelles

- Match

```
>>> command = 'Hello, World!'
>>> match command:
...         case 'Hello, World!':
...             print('Hello to you too!')
...         case 'Goodbye, World!':
...             print('See you later')
...         case other:
...             print('No match found')

Hello to you too!
```

```
>>> command = 'remove file1.txt file2.jpg file3.pdf’
>>> match command.split():
...         case ['show']:
...             print('List all files and directories: ')
...             # code to list files
...         case ['remove', *files]:
...             print('Removing files: {}'.format(files))
...             # code to remove files
...         case _:
...             print('Command not recognized’)

Removing files: ['file1.txt', 'file2.jpg', 'file3.pdf']
```

Les instructions répétitives

- Permet d'exécuter un bloc d'instructions plusieurs fois de suite selon une ou plusieurs conditions.

- Il existe différents types de boucles :
  - Boucles avec compteur
  - Boucles conditionnelles
  - Boucles de parcours

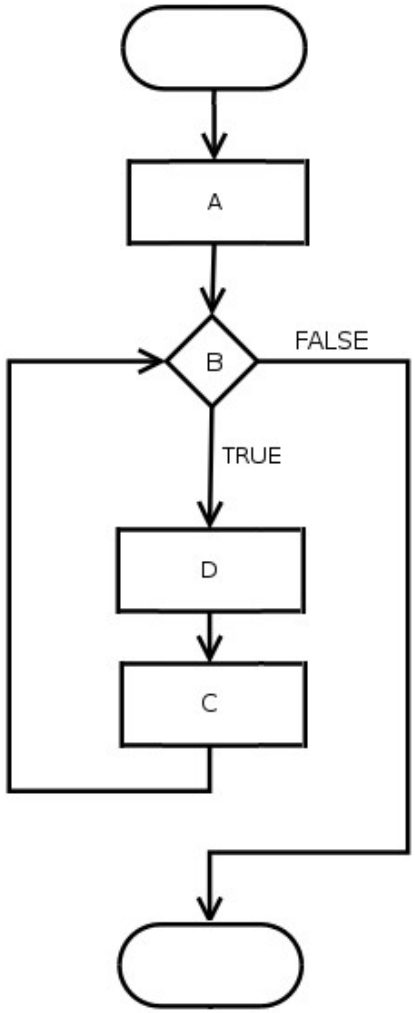
Les boucles avec compteur

- Les boucles *for*

- Variable qui prend des valeurs successives sur un intervalle.

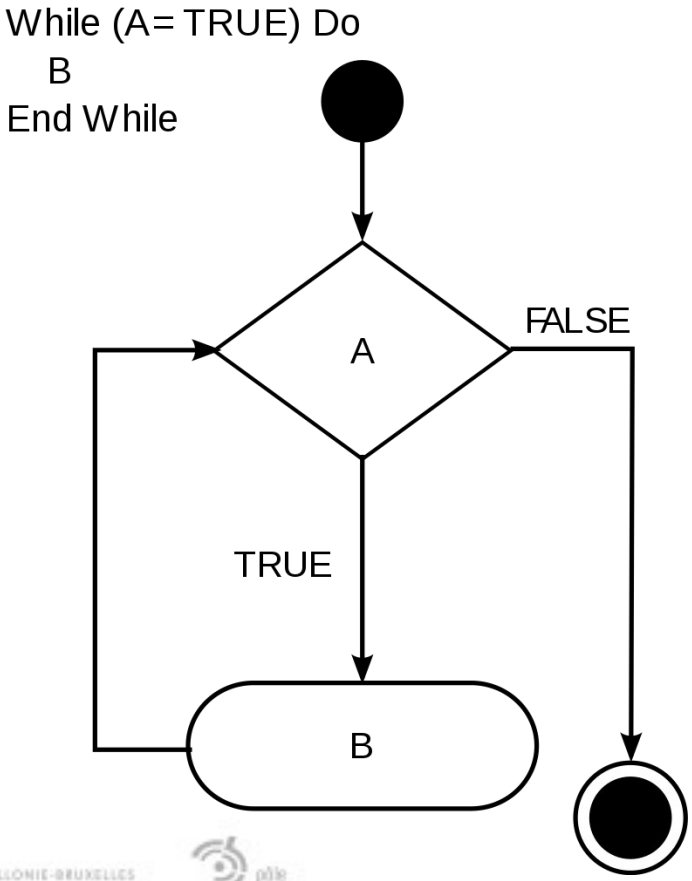
```
int t[5];
for (int i = 0; i < 5; i++)
{
   t[i] = i;
}
```

for(A;B;C)
D;

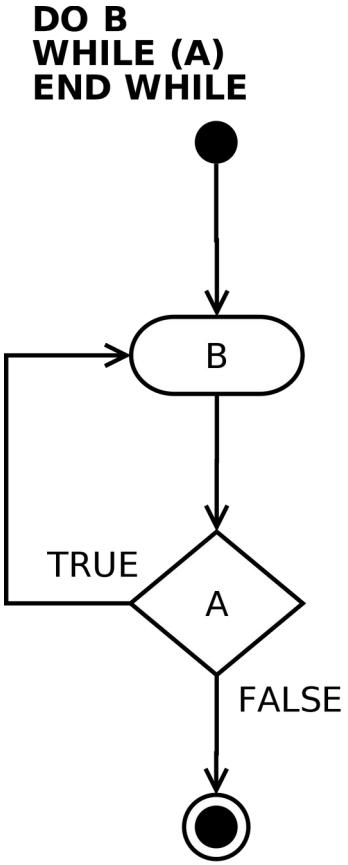


Les boucles conditionnelles

- Les boucles *while*



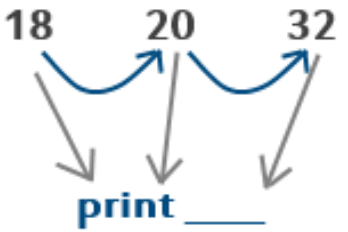
- Les boucles *do…while*



Les boucles de parcours

- Les boucles *foreach*
- Parcourir les objets d'une collection.

```
seq = [18, 20, 32]
for each x of seq
    print x
end
```




```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
    print(x)
```

Exercices

```
x=4;
if (x%2==0) {
     x/=2;
     if (x%3==0) {
          x/=3;
     }
     else {
          x*=3;
     }
}
printf("%d",x);
```

Exercices

```
x=55;
if ((x%3==0) && (x%5==0)) {
    print("Check1");
}
else if ((x%3==0) || (x%5==0))
{
    print("Check2");
}
```

Exercices

```
i=2;
do {
     i*=2;
} while ( i<10 );
printf("%d", i);
```

Exercices

```
for (i=1 ; i<=10 ; i+=2) {
    print("%d", i);
}
```

Chapitre 3

Résolution de problèmes

Poser un problème

- Trouver le chemin le plus court entre la HEH et le marché aux herbes
- Calculer la factorielle de 10
- J’ai 10€ et j’en dépense 2. Calculer combien il me reste d’€.

Poser un problème

- Trouver le chemin le plus court entre 2 endroits *a* et *b* dans Mons
- Calculer la factorielle du nombre réel positif *p*
- J'ai *x* € et j'en dépense *y*. Calculer combien il me reste d'€

Spécification d'un problème

- Paramètres d'entrée
  - → Caractériser une instance du problème
  - → Variables de l'énoncé

- Pré-conditions
  - → Conditions à respecter pour que le problème aie un sens

Spécification d'un problème

- Paramètres de sortie
  - → Caractérise la solution à une instance du problème
  - → Eléments de la réponse

- Post-conditions
  - → Conditions à respecter par les paramètres d'entrée et de sortie

Spécification d'un problème

- Exemples :

Entrée a – une adresse
Entrée b – une adresse
Pré-condition – a et b sont à Mons
Sortie c – un chemin
Post-condition – c relie a et b
longueur(c) < longueur(c')

Spécification d'un problème

- Exemples :

  Entrée a – un nombre
  Pré-condition – a est un nombre réel positif
  Sortie b – la factorielle
  Post-condition – b = a!

Formalisation

Etapes

Etape 1 : Démarrer

Etape 2 : Lire nombre

Etape 3 : Mettre fact à 1 et i à 1

Etape 4 : Vérifier que i <= nombre. Si faux aller à l'étape 7

Etape 5 : fact = fact*i

Etape 6 : i = i+1 et retourner à l'étape 4

Etape 7 : Afficher fact

Etape 8 : Quitter

Schéma