Intro C++

UE15
Informatique
appliquée

Frédéric Pluquet
pluquetf@helha.be

2018-2019

Premier programme en C++

C++

```cpp
// A Hello World! program in C++
#include <iostream>
using namespace std;

int main() {
  cout << "Hello world" << endl;
  return 0;
}
```

C#

```csharp
// A Hello World! program in C#.
using System;
namespace HelloWorld
{
    class Hello
    {
        static void Main()
        {
            Console.WriteLine("Hello World!");

            // Keep the console window open in debug mode.
            Console.WriteLine("Press any key to exit.");
            Console.ReadKey();
        }
    }
}
```

Compilation

0_hello world.cpp

```C++
// A Hello World! program in C++
#include <iostream>
using namespace std;

int main() {
  cout << "Hello world" << endl;
  return 0;
}
```

```
→  codes g++ 0_hello\ world.cpp -o 0_hello\ world.o --std=c++11
→  codes ./0_hello\ world.o
Hello world
→  codes
```

Paradigmes

Procédural

```cpp
// A Hello World! program in C++
#include <iostream>
using namespace std;

int main() {
  cout << "Hello world" << endl;
  return 0;
}
```

Orienté objet

```cpp
class Point
{
  double x, y;

  public:
    Point(int x, int y)
    {
      this->x = x;
      this->y = y;
    }

    double getX()
    {
      return x;
    }

    double getY()
    {
      return y;
    }
}
```

Pour commencer en C++…

- Commentaires
- Variables, types et constantes
- Références
- Les fonctions
  - Définition
  - Paramètres par défaut
  - Surcharge
  - Inline
- Résolution de portée

Commentaires

- Commentaire de fin de ligne:

```
//Ceci est un commentaire
```

- Commentaire sur plusieurs lignes :

```
/* Ceci est un commentaire qui peut se tenir
sur  plusieurs lignes */
```

- Commentaire sur plusieurs lignes avec macro:

```
#ifdef un_commentaire
quelque chose à mettre en commentaires  sur plusieurs
lignes aussi!
#endif
```

Déclaration des variables

Les variables peuvent être déclarées n’importe où
pour autant qu’elles soient déclarées avant d’être utilisées.

Types de base

- Les types de base: `char, int, float, double, void, bool, enum`.

- Les variables de type `bool` peuvent avoir 2 valeurs :
  - 1 (true)
  - 0 (false)

- Les énumérations peuvent devenir un type. La syntaxe de *enum* est :

```
enum [ type_enum]{nom_constante[=valeur],...}
```

Déclaration de constantes

Pour les constantes:

const *type* identificateur = *valeur_initiale*;

Variables et constantes

Qu'affiche le programme suivant ?

```
1   // #include <stdio.h>
2   #include <iostream>
3   using namespace std;
4
5   int main() {
6     const int n = 40;
7     // n = 12; // erreur à la compilation
8     for (int i=0;i<n;i++) {
9       int j = 25;
10      i += j;
11      cout << "i vaut: " << i << endl;
12    }
13    return 0;
14  }
```

Bool

Qu'affiche le programme suivant ?

```cpp
#include <iostream>

int main() {
  int n = 40;
  bool isFound = false;
  for (int i=0; i<n && !isFound;i++) {
    int j = 25;
    i += j;
    std::cout << "i vaut: " << i << std::endl;
    if (i > 20) {
      isFound = true;
    }
  }
  return 0;
}
```

Enum

Qu'affiche le programme suivant ?

```cpp
#include <iostream>

int main() {
  enum Color { red, green, blue };
  Color r = red;
  switch(r)
  {
      case red  : std::cout << "red\n";   break;
      case green: std::cout << "green\n"; break;
      case blue : std::cout << "blue\n";  break;
  }
  return 0;
}
```

Références

- Une référence est un nouveau nom donné à une variable existante (a un rôle d'alias)

- Une référence ne peut pas être vide (il faut l'initialiser directement)

- Pour définir une référence:

```
type &nomReference=variableAReferencer
```

Références

Qu'affiche le programme suivant ?

```cpp
#include <iostream>
using namespace std;

int main() {
  int i=0, j=20;
  int& valeur = i;

  cout << "i: " << i << " valeur: " << valeur << endl;
  i = 10;

  cout << "i: " << i << " valeur: " << valeur << endl;
  valeur = j;
  valeur += 5;

  cout << "i: " << i << " valeur: " << valeur << endl;
  return 0;
}
```

Les fonctions - main

- `main` est une fonction qui est appelée automatiquement à l'exécution du programme
- Ses seuls prototypes sont:
  - `int main()`
  - `int main(int argc, char* argv[])`
  - `int main(int argc, char* argv[],char* envp[])`

Les fonctions - main

Exercice: Qu'affiche le programme suivant si on passe les paramètres en ligne de commande: `1 et 1 font 2` ?

```
1   #include <iostream>
2   using namespace std;
3
4   int main(int argc, char* argv[],char* envp[]){
5
6     cout << "Le programme " << argv[0] << " a " << argc << " arguments" << endl;
7
8     for (int i=0;i<argc;i++)
9       cout << argv[i] << endl;
10
11    cout << "Variables d'environnement:\n" << endl;
12
13    for(int i=0;envp[i];i++)
14      cout << envp[i] << endl;
15
16    return 0;
17  }
```

Les fonctions – Déclaration et définition

- La déclaration des prototypes de fonctions est obligatoire avant leur utilisation
- Syntaxe:

```
type nom_fonction(type_var1 var1[=valeur1],…,type_varn varn[=valeurn]);
```

Les fonctions - Paramètres par défaut

C++ permet d'avoir des fonctions à nombre variable de paramètres en donnant des valeurs par défaut aux paramètres.

Les fonctions peuvent être appelées avec un nombre de paramètres inférieurs à celui de la déclaration.

Les fonctions - Paramètres par défaut

Qu'affiche le programme suivant ?

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

const bool STOP = true;
void afficheMessage(const char *msg="Avertissement", bool stop=false) {
  printf("%s\n",msg);
  if (stop)
    printf("STOP\n");
}
int main(){
  afficheMessage("erreur fatale",STOP);
  afficheMessage("fausse manoeuvre",0) ;
  afficheMessage("fausse manoeuvre");
  afficheMessage();
  return 0;
}
```

Les fonctions - Surcharge

Plusieurs méthodes ayant :

le même nom,
mais des signatures différentes,
pour un même programme

Les fonctions - Surcharge

Exercice : Quelles sont les fonctions appelées dans le programme suivant, s'il n'y a pas d'erreur ?

```c
#include <stdio.h>

double exemple(double x, double w){ // configuration -A-
  printf("configuration -A-:%f %f \n",x,w);
  return x;
}
int exemple(int x, int w){ // configuration -B-
  printf("configuration -B-:%d %d \n",x,w);
  return x;
}
int main() {
  int b=200,v;
  double a=100.5, w;
  // essayer sans les cast
  v = exemple((double)b, a);
  w = exemple(b, (int) a);
  w = exemple(b, b);
  return 0;
}
```

Les fonctions - inline

Les fonctions en ligne permettent d'utiliser de petites fonctions lorsque, pour des raisons d'efficacité, le programmeur ne veut pas appeler une fonction pour effectuer les quelques instructions qui implémentent une opération.

```
inline int min(int a, int b){
  return ((a<b)?a:b);
}
…
int m=min(i,j); //remplacé par m=((i<j)?i:j); à la compilation
```

Résolution de portée

Il est possible d'exploiter la variable globale en faisant précéder cette variable du double symbole :: (opérateur de résolution de portée).

L'opérateur :: fait toujours référence à la variable globale.

Résolution de portée: exercice

Qu'affiche le programme suivant ?

```
#include <iostream>
using namespace std;

int x=10 ;

int main(void) {
  cout << x << endl;
  char x = 'b';
  cout << x << endl;
  cout << ::x << endl;
  {
    cout << "sous-bloc" << endl;
    double x = 2.15;
    cout << x << endl;
    cout << ::x << endl;
  }
  return 0 ;
}
```

Des questions ?