les éléments essentiels pour commencer

christophe.labourdette(at)cmla.ens-cachan.fr

Septembre 2016

Un minimum

```
int main()
    {
      // un programme vide
      return 0;
    }
```

Il doit toujours y avoir un main. l'instruction return permet de finir la fonction en donnant la valeur de retour, ici 0.

entête

On ajoute en général l'inclusions des fichiers d'entêtes pour le préprocesseur au tout début du fichier.

```
#include <iostream>
#include "perso.hpp"
```

On note que les entêtes systèmes sont encadrées par < et > alors que les entêtes de l'utilisateur sont entres ″.

variables élémentaires

Toutes les variables doivent être déclarées.

```
int i,j=1;
char a,b='g';
float x,y=-3.2;
double v,w=0.32e-78;
bool oui,non=false;
```

- Elle peuvent être initialisées (ou pas) lors de leur déclaration
- Une variable non initialisée peut contenir n'importe quel valeur
- Les variables peuvent être déclarées n'importe ou
- La portée est limitée au bloc dans lequel elles sont déclarées

string

La librairie standard comprend une classe permettant de manipuler des chaînes de caractères.

```
std::string nom;
std::string prenom;
std::cout <<"Donnez votre pr nom :";
std::cin >> prenom;
std::cout <<"Donner votre nom :";
std::cin >> nom;
```

vector

La **STL** permet d'utiliser la classe vector. Comme son nom l'indique, il s'agit d'un vecteur typé.

```
std::vector<int> I={7, 9, 2, 5, 7, 8};
std::vector<double> X={1.0, 2.0, 3.0, 4.0};
std::vector<long> IL(151);
std::vector<float> XF(17);
std::vector<string> EC;
```

operateurs

Il est possible d'utiliser des opérateurs pour faire des calculs ou agir sur des objets. L'action d'un opérateur dépend de l'objet sur lequel il porte.

```
int i1=3, i2=7, i3;
i3 = i1 + i2;
std::string a="prenom";
std::string b="nom";
std::string blanc=" ";
std::string c = a+blanc+b;
std::cout << "i3 = " << i3;
std::cout << " c = " << c << std::endl;
```

Avec les entiers l'addition est l'opération usuelle, avec les chaines de caractères on concatène.

Calculs

Avec les short, int , long, float , double on peut utiliser les opérateurs suivants :
+ , − , ∗ , / , ++ , −− , += n , −= n , ∗= , /=
Avec les différents entiers, il existe de plus % , %=
Toutes les fonctions classiques (y compris les puissances ne sont pas dans le langage et sont des fonctions externes.

Opérations

- a.b : l'élément b de l'objet a
- a[b] : l'élément dans l'objet d'indice b
- a++ : incrémente a et renvoie la valeur originale
- a−− : décrémente a et renvoie la valeur originale
- ++a : incrémente a et renvoie la valeur incrémentée
- −−a : décrémente a et renvoie la valeur décrémentée
- a ∗ b : le produit de a et b
- a / b : la division de a par b
- a % b : le reste de la division si a et b sont des entiers
- a + b : la somme de a et b
- a − b : la soustraction de a par b

Opérations (suite)

- a opb b : avec *opb* un opérateur $<<$ ou $>>$, décalage de bits
- a opr b : avec *opr* un opérateur relationnel (<,<=,>,>=), booléen
- a==b : booléen indiquant si a est égal à b
- a!=b : booléen indiquant si a est différent de b
- a && b : booléen vrai si a et b sont vrai
- a || b : booléen vrai si a ou b sont vrai
- a = b : assignation
- a op= b : equivalent à a = a op b où op est par exemple +,$-$,$*$,/,%,<<,>>)
- a ? b : c : renvoie b si a est vrai, c sinon

conditions

Il est possible de n'exécuter un bloc d'instructions que sous certaines conditions :

```
int age=0;
std::cout << "Entrez votre age : ";
std::cin >> age;
if (age < 18)
{
  std::cout << "Vous n'avez pas la majorite.\n";
  exit 1;
}
else
{
...
}
```

boucles

la boucle la plus simple et la plus facile à utiliser est :

```
int i,j=0;
for (i=0;i<12;i++)
{
   j += i * (j + i + 1);
   std::cout<< "i=" <<i<<",j="<<j<< std::endl;
}
```

En général

```
for(initialisation;condition de fin;increment)
```

Mais il y a également while et until.

Opérations logiques

La valeur d'une expression logique est booléenne elle est donc soit true soit false.
On peut utiliser les opérateurs suivants :

- le "ou" logique ||
- le "et" logique &&
- le "non" logique !
- les opérations de comparaison <, <=, >, >=, ==, !=

Attention la conversion entre booléen et entier n'est pas intuitive :

- false est 0,
- vrai est n'importe quel nombre non nul,
- la valeur 1 est bien entendu la valeur préférée quand c'est possible.

fonction

Il est bien entendu possible de construire des fonctions différentes du int main().
En voici un squelette :

```
typeDeRetour nomDeLaFonction(listeDeParametres)
  {
    typeDeRetour aRenvoyer;
    ...
    return aRenvoyer;
  }
```

Attention les paramètres sont passés par valeur.