IFT159
Analyse et programmation
Chapitre 2 — Éléments de C++

Gabriel Girard

Département d'informatique

UNIVERSITÉ DE SHERBROOKE

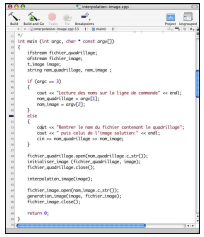
8 septembre 2009

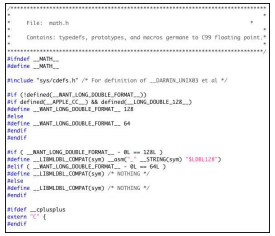
Éléments de C++

Qu'est-ce qu'un programme C++

un programme est composé

- Votre code
- Du code existant (Bibliothèques)





Entête

```
                /*                                  */
(1) ⟶           /* Identification du programme      */
                /*                                  */
(2) ⟶           Directive #include
(3) ⟶           using namespace std ;
```

La fonction main

(4) ⟶ `int main()`
`{`
(5) ⟶ *définitions/déclarations des constantes*
(6) ⟶ *définitions/déclarations des variables*

(7) ⟶ *énoncés exécutables*

(8) ⟶ `return 0 ;`
`}`

(1) Commentaires

- Deux formes
  - //
  - /* . . . */
- C'est de la documentation (sert à la maintenance)
- Utilisés éventuellement par d'autres outils
- Ignorés par le compilateur
- Normes départementales à respecter
- Documentation et normes : 30 % de la note.


```
/**
  Fichier : consommation.cpp

  Calcul de consommation d'essence d'une voiture.

  Auteur : Gerard Houdeville
  Date de derniere modification :  19 aout 2006 (derniere version)
  Date de création :  8 janvier 1996 (création)
  Version 1.4 : 19 aout 2006, mise aux normes, Benoit Fraikin
  Version 1.3 : 3 jan. 2004, ajout documentation, Benoit Fraikin
  Version 1.2 : 16 juil. 2003, modification legere, Gabriel Girard
  Version 1.1 : 8 jan. 1996, modification legere, Gerard Houdeville

  Entrées :
    (clavier) distance parcourue (reel positif non nul)
    (clavier) volume utilise (reel positif non nul)
  Sorties
    (ecran) consommation (reel positif)

  .... description detaillee.............
*/
```


(2) Directives au compilateur

- Directive `#include`

```
#include <nom_du_fichier>
#include "nom_du_fichier"
```

- Indique au compilateur d'insérer du code existant
- Exemple :

```
#include <cmath>
#include <iostream>
#include <string>
```

(4) Fonctions

- Toute manipulation utilise une fonction en C++
- Une fonction réalise une tâche précise
- Pour IFT159, cette tâche sera décrite dans un module lors de la conception
- Tout programme possède une fonction `main`

(5-6) Définitions (constantes et variables)

- Assignent un nom aux données
- Définissent l'ensemble des valeurs que peut recevoir une variable ou une constante (**type**)
- Noms significatifs
- Noms composés de lettres, chiffres et «_»
- ATTENTION aux noms réservés
- Normes
  - Mots séparés par «_» ou majuscules ;
  - Une variable débute par une minuscule ;
  - Constantes en majuscules ;
  - Pas de lettres accentuées.

(5) Définition des constantes

const *type NOM_CONSTANTE = valeur* ;

- `const float PI = 3.1416 ;`
- `const int NB_MOIS = 12 ;`

(6) Définition des variables

type liste_d'identificateurs ;

- `int jour, mois, annee ;`
- `char initiale_prenom ;`
- `float abcisse,ordonnee ;`
- `int nbJour, nbHeure, nbSeconde ;`

```
int main()
{
  // Declaration des constantes
  const float LITRE_A_GALLON = 1/4.5;    //constante de conversion
  const float KILOMETRE_A_MILLE = 1/1.6; //constante de conversion

  // Declaration des variables locales
  float distance_kms ;             //distance parcourue en kms
  float volume_litres ;            //quantite d'essence en litres
  float distance_milles ;          //distance parcourue en milles
  float volume_gallons ;           //quantite d'essence en gallons
  float consommation ;             //consommation d'essence

  ...

}
```


(7) Énoncés exécutables

- Énoncé d'affectation
- Énoncé de lecture
- Énoncé d'écriture

Énoncé d'affectation

Permet de modifier la valeur d'une variable et d'entreprendre certaines actions :

identificateur_variable = *expression_mathématique*;

- `compteur = 0 ;`
- `total = total + (prix * taux_taxe) ;`
- `somme = somme + 1 ;`

Énoncé de lecture

- `cin >> note ;`
- `cin >> mois >> jour >> annee ;`
- Opérateur : `>>`
- Médium : `cin`

Énoncé d'écriture

- `cout << note ;`
- `cout << "Le nombre d'etudiants est " << nbr_etu << endl ;`
- `cout << "Bienvenue " << endl << endl << "au cours" ;`
- Opérateur : <<
- Médium : `cout`


```
// 1. lecture distance et quantite sur entree standard (clavier)
cout << endl << "Donner la distance parcourue en kilometres: " ;
cin >> distance_kms ;
cout << "Donner la quantite d'essence en litres: " ;
cin >> volume_litres ;

// 2. Conversions et calcul de la consommation
//   2.1 Conversion de la distance en milles
distance_milles = distance_kms * KILOMETRE_A_MILLE ;
//   2.2 Conversion du volume en gallons
volume_gallons = volume_litres * LITRE_A_GALLON ;
//   2.3 Calcul de la consommation en milles/gallons
consommation = distance_milles / volume_gallons ;

// 3. Affichage de la consommation
cout << endl << "La consommation en milles au gallon est : " ;
cout << consommation << endl ;

// Valeur de retour
return 0;
}
```


Abstraction

- Simplification ou généralisation d'un concept ou d'un objet
- Abstraction procédurale
- Abstraction de données (types abstraits)
  - Types de données de base
  - Types de données pré-définies (bibliothèques)
  - Vos types de données (classes)

Types de données de base : **float**

- Abstraction de l'ensemble $\mathbb{Q}$ (rationnels – réels par abus de langage) compris entre deux bornes MIN et MAX
- Opérations : $+, -, *$ et $/$.
- `const float PI = 3.1416 ;`
- `const float TEST = 2.1e-3 ;`
- `float result ;`
- `result = PI * TEST / 0.04 ;`

Types de données de base : **int**

- Abstraction de l'ensemble $\mathbb{Z}$ (entiers) entre MIN et MAX
- Opérations : $+, -, *, /$ et %
- `const int TEST1 = 4 ;`
- `int test2, test3, test4 ;`
- `test2 = TEST1 * 2 + 5 / 6 % 2`

Types de données de base : **char**

- Ensemble de symboles incluant les lettres minuscules et majuscules, les chiffres et des symboles spéciaux.
- Opérations : ? ? ?
- `const char INITIALE = 'p' ;`
- `char car1 ;`
- `cin >> car1 ;`
- `cout << "La lettre initiale est "`
  `<< INITIALE ;`

Types de données de base : **bool**

- Valeurs de vérité : vrai (*true*) ou faux (*false*)
- Opérations : `&&`, `||` et `!`
- Le nom de la donnée doit contenir un verbe
- `bool est_un_echec, continuer ;`
- `est_un_echec = true ;`
- `continuer = !est_un_echec && (x > 0) ;`

Types de données ajoutés : **string**

- Fourni dans une librairie
- `#include <string>`
- Opérations : ? ?
- `const string PRENOM = "Arthur" ;`
- `string nom ;`
- `cin >> nom ;`
- `cout << PRENOM << nom ;`

Types d'erreurs

- Erreurs de syntaxe : la plus simple à corriger
- Erreurs à l'édition des liens
- Erreurs à l'exécution
- Erreurs de logique : la plus compliquée car peut remonter très haut dans le développement (conception ou analyse)

Exemples d'erreurs

- F16 - l'avion se retournait sur le dos lorsqu'il traversait l'équateur
- F16 - une fois sur le dos l'ordinateur ne pouvait décider entre rouler sur la gauche ou sur la droite → interblocage
- Une dame de 104 ans a reçu une invitation pour entrer en maternelle
- Un homme de 101 ans a vu son assurance tripler car il a été classé comme adolescent (moins de 20 ans).