IFT159
Analyse et programmation
Chapitre 4 — Structures sélectives

Gabriel Girard

Département d'informatique

UNIVERSITÉ DE SHERBROOKE

23 septembre 2008

Chapitre 4 — Structures sélectives

Nécessité de la sélection

Exemple de besoin

Exemple 1 : trouver la valeur absolue d'un nombre.

Nécessité de la sélection

Exemple de besoin

Exemple 2 : déterminer si la valeur d'un nombre est paire ou impaire.

Opérateurs logiques

- Deux types d'opérateurs : relation et logiques ;
- Tableau des opérateurs :

$<$	Plus petit	relation	binaire
$<=$	Plus petit ou égal	relation	binaire
$>$	Plus grand	relation	binaire
$>=$	Plus grand ou égal	relation	binaire
$==$	égal	relation	binaire
$!=$	Différent	relation	binaire
&&	Et logique	logique	binaire
$\|\|$	Ou logique	logique	binaire
!	Non logique	logique	unaire

Opérateurs logiques et prédicat

- Un opérateur de relation prend (en général) deux variables d'un certain type et renvoie une valeur de vérité.
- Un opérateur de logique prend (en général) deux variables qui sont des valeurs de vérité et renvoie une valeur de vérité.
- Une fonction qui renvoie une valeur de vérité est appelée un *prédicat*.
- Son nom doit contenir un verbe :

```
bool est_une_annee_bisextile(int);
```


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);`

Opérateurs de relation

```
int nb1, nb2;
float val;
char lettre;

nb1 < 10;
nb1 == nb2;
20.0 <= val;
lettre != 'a';
((nb1 + nb2) / 2) > 50;
```

Opérateurs logiques

&&	V	F
V	V	F
F	F	F

||	V	F
V	V	V
F	V	F

!	V	F
	F	V

Exemple

```
(nb1 < 10) && (nb2 > 50)
(lettre < 'z') || (lettre > 'a')
!(( val < 62.25) && (nb1 >5))
```


Règles de préscéance

+ prioritaire	
	! , +(un.), - (un.)
	* , / , %
	+ , -
	<, <=, >, >=
	== , !=
	&&
	||
	=
- prioritaire	

Exemple

```
        x < min + max   ≡   x < (min + max)
min <= x && x <= max    ≡   (min <= x) && (x <= max)
```


Relations entre les caractères

- $'0' <' 1' < ... <' 9'$
- $'A' <' B' < ... <' Z'$
- $'a' <' b' < ... <' z'$
- chiffre $<$ majuscule $<$ minuscule

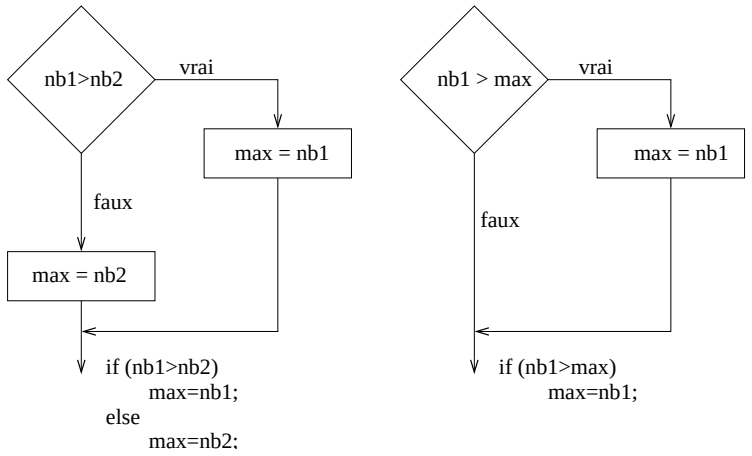
Exercices

- Quelle est la valeur de l'expression
  `(lettre < 'z') || (lettre > 'a')`
  sachant que lettre contient une lettre minuscule ?
- Expression pour tester si x est compris dans l'interval]-50,+50].
- Expression pour tester si x est en dehors de l'interval [-50,+50[.
- Expression pour tester si x est compris dans l'interval]-50,-20] ou dans l'interval]20,50].

Énoncé if : syntaxe

- `if ( condition )`
  enonce_vrai
- `if ( condition )`
  enonce_vrai
  `else`
  enonce_faux

Énoncé if



Énoncé composé

- Énoncé simple vs énoncé composé
- Un énoncé composé (bloc) est une séquence d'énoncés entre { }.

Énoncé composé : exemple

```
if (val1 > val2)
{
    int temp ;
    temp = val1 ;
    val1 = val2 ;
    val2 = temp ;
}
```

Énoncé if : syntaxe *suggéré*

- ```
  if ( condition )
  {
        enonce_vrai
  }
  ```
- ```
  if ( condition )
  {
        enonce_vrai
  }
  else
  {
        enonce_faux
  }
  ```


Énoncés if emboîtés

- La partie *enonce_vrai* et *enonce_faux* d'un if peuvent être n'importe quel énoncé connu, donc en particulier d'autres énoncés if.
- On peut donc construire des énoncés à plusieurs alternatives (if emboîtés).

Énoncés if emboîtés : exemple

Dans un ensemble de valeurs entières, on désire connaître le nombre de valeurs négatives, positives et nulles.

```
if ( nombre > 0 )
    compt_pos = compt_pos + 1 ;
else
    if ( nombre == 0 )
        compt_nul = compt_nul + 1 ;
    else
        compt_neg = compt_neg + 1 ;
```


Énoncés if emboîtés : exemple

```
if ( nombre > 0 )
{
    compt_pos = compt_pos + 1 ;
}
else
{
    if ( nombre == 0 )
    {
        compt_nul = compt_nul + 1 ;
    }
    else
    {
        compt_neg = compt_neg + 1 ;
    }
}
```

Lisibilité des `if` emboîtés

- Comment écrire une multitude de `if` emboîtés ?
- Problème de lisibilité accru
- $\Longrightarrow$ Risque d'erreur supplémentaire

Lisibilité des "if" emboîtés

La côte A est attribuée pour une évaluation totale supérieur à 85,
B si supérieur à 70,
C si supérieur à 55,
D si supérieur à 40,
E dans les autres cas.

Écrire la portion de code réalisant ceci.

Lisibilité des "if" emboîtés

```
if (eval_tot > 85)
        cout  << 'A' ;
else if (eval_tot > 70)
        cout  << 'B' ;
else if (eval_tot > 55)
        cout << 'C' ;
else if (eval_tot > 40)
        cout << 'D' ;
else
        cout << 'E' ;
```


Lisibilité des "if" emboîtés

```
if (eval_tot > 85)
{
        cout  << 'A' ;
}
else if (eval_tot > 70)
{
        cout  << 'B' ;
}
else if (eval_tot > 55)
{
        cout << 'C' ;
}
else if (eval_tot > 40)
{
        cout << 'D' ;
}
else
{
        cout << 'E' ;
}
```

L'énoncé « switch »

- Cet énoncé permet de faire des sélections à alternatives multiples.
- La sélection doit être basée sur une variable ou une expression de type obligatoirement entier ou caractère (sélecteur).
- Le sélecteur sera ensuite comparé à des valeurs particulières constantes (des étiquettes de cas).
- Le type du sélecteur et celui des étiquettes doivent être identiques.
- Ceci est interprété comme suit : « Au cas où le sélecteur vaut une valeur particulière alors effectuer... ».

L'énoncé « switch » :exemple

Simulons un guichet automatique où les transactions possibles sont :

le dépot	symbolisé par la lettre « d » ;
le retrait	le retrait symbolisé par la lettre « r » ;
le solde	symbolisé par la lettre « s » ;
le transfert	symbolisé par la lettre « t » et
le paiement	symbolisé par la lettre « p ».

L'énoncé « switch » :exemple

```
switch (code_transaction)
{
      case 'd': case 'D': depot() ;
                break ;
      case 'r': case 'R': retrait() ;
                break ;
      case 's': case 'S' : solde() ;
                break ;
      case 't': case 'T' : transfert() ;
                break ;
      case 'p': case 'P' : paiement() ;
                break ;
      default : cout << "Erreur de code" << endl ;
}
```


Cas général du « switch »

```
switch (selecteur) // type int ou char
{
        case et1:
                  enonces_1;
                  break ;
        case et2:
                  enonces_2;
                  break ;
        case et3:
                  enonces_3;
                  break ;
        default : enonces_sinon ;
}
```

Diagramme structurel et flots de données

- On peut ajouter de l'information dans le diagramme structurel.
- On ajoute le flot de circulation des données entre les modules du diagramme.

Diagramme structurel et flots de données

- Exemple : Écrire un programme calculant le salaire brut et le salaire net d'un employé connaissant son nombre d'heures travaillées et le taux horaire auquel il travaille. Des déductions à la source sont imposées comme suit : $25 pour tout salaire inférieur à $250, 10% pour tout gain compris entre $250 et $750 et pour les gains supérieurs à $750, 15% sur la tranche excédentaire des $750. On veut que la personne en charge de la paie soit informée du fonctionnement du programme.

Présentation des sorties

- On peut améliorer la présentation de la sortie (fixer le nombre de décimales, exiger la notation décimale ou la notation scientifique...)
- `#include <iomanip>`
- Détails dans le manuel.

Jeu d'essais

- Comment vérifier l'exactitude des résultats.
- Problème : Lire trois lettres et afficher celle qui est alphabétiquement la plus petite.

Jeu d'essais

Jeu de tests : Pour être assuré que l'algorithme fonctionne (avant l'implmentation) et que le programme fonctionne il faut par traçage manuel (faire en exercice) et par jeu d'essai :

- Placer la plus petite lettre en 1er, puis 2ième puis 3ième position.
- Tester les cas limites :
  - la + petite apparaît deux fois.
  - la + petite apparaît trois fois.

Modifiabilité

- On peut modifier une solution existante (maintenance).
- On peut avoir un nouveau problème qui s'avère très similaire à une solution existante.
- La solution existante doit pouvoir être adaptée.

Modifiabilité : Exemple

Écrire un programme calculant le salaire brut et le salaire net d'un employé connaissant son nombre d'heures travaillées et le taux horaire auquel il travaille. Cet employé peut avoir fait des heures supplémentaires qui sont payées une fois et demi le tarif normal. Normalement un employé travaille 35.5 heures. Des déductions à la source sont imposées comme suit : $25 pour tout salaire inférieur à $250, 10% pour tout gain compris entre $250 et $750 et pour les gains supérieurs à $750 15% sur la tranche excédentaire des $750. On veut que la personne en charge de la paie soit informée du fonctionnement du programme.

Solution par analogie

- Il arrive parfois qu'un nouveau problème soit similaire à un ancien déjà solutionné mais dans un autre domaine et avec une formulation différente.
- Exemple : Une compagnie d'assurance accorde, en fin d'année, une ristourne à tous ses détenteurs de polices. Le taux de la ristourne est de 3.5% de la prime pour tous, et une ristourne additionnelle de 1% de la prime est accordée à tous ceux qui n'ont pas fait de réclamation durant l'année. Faire l'analyse, la conception et l'implantation de ce problème.