IFT159
Analyse et programmation
Chapitre 4 — Structures sélectives

Gabriel Girard

Département d'informatique
UNIVERSITÉ DE SHERBROOKE

22 décembre 2015

Chapitre 4 — Structures sélectives

UNIVERSITÉ DE SHERBROOKE

Structures sélectives

UNIVERSITÉ DE SHERBROOKE

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.

Structures sélectives

Opérateurs logiques

- Deux types d'opérateurs : relation et logiques ;

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
&& (and)	Et logique	logique	binaire
|| (or)	Ou logique	logique	binaire
! (not)	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.

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.

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*.

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*)

Types de données de base
`bool`

- Valeurs de vérité : vrai (*true*) ou faux (*false*)
- Opérations : `&&` (and), `||` (or) et `!` (not)

Types de données de base
`bool`

- Valeurs de vérité : vrai (*true*) ou faux (*false*)
- Opérations : `&&` (and), `||` (or) et `!` (not)
- Le nom de la donnée doit contenir un verbe

Types de données de base
`bool`

- Valeurs de vérité : vrai (*true*) ou faux (*false*)
- Opérations : `&&` (and), `||` (or) et `!` (not)
- Le nom de la donnée doit contenir un verbe
- `bool est_un_echec, continuer;`

Types de données de base
`bool`

- Valeurs de vérité : vrai (*true*) ou faux (*false*)
- Opérations : `&&` (and), `||` (or) et `!` (not)
- Le nom de la donnée doit contenir un verbe
- `bool est_un_echec, continuer;`
- `est_un_echec = true;`

Types de données de base
`bool`

- Valeurs de vérité : vrai (*true*) ou faux (*false*)
- Opérations : `&&` (and), `||` (or) et `!` (not)
- 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 de base

bool

Opérateurs de relation

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

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

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'

Relations entre les caractères

- '0' < '1' < ... < '9'
- 'A' < 'B' < ... < 'Z'

Relations entre les caractères

- '0' < '1' < ... < '9'
- 'A' < 'B' < ... < 'Z'
- 'a' < 'b' < ... < 'z'

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 ?

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].

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[.

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].

Structures sélectives

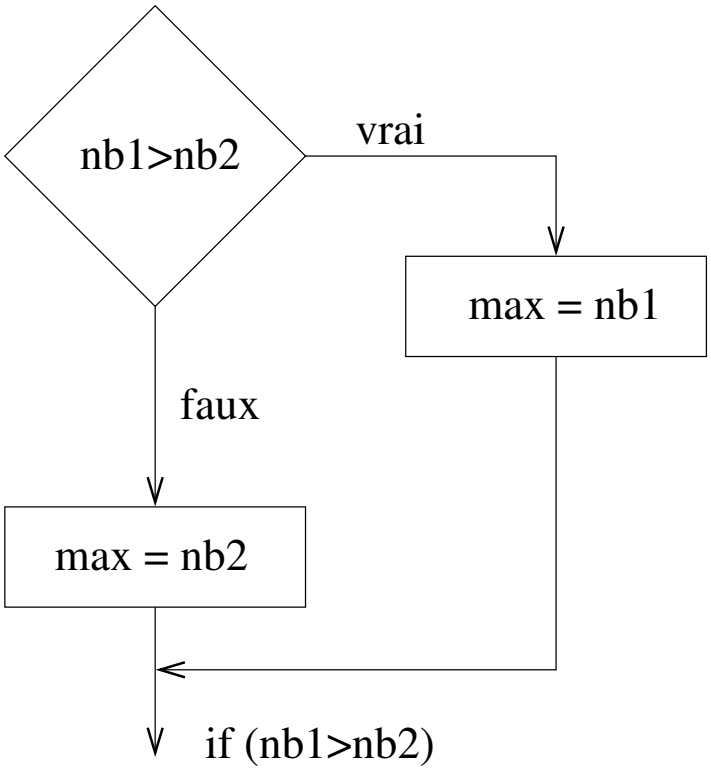
Énoncé `if` : syntaxe

- `if ( condition )`
  `enonce_vrai`

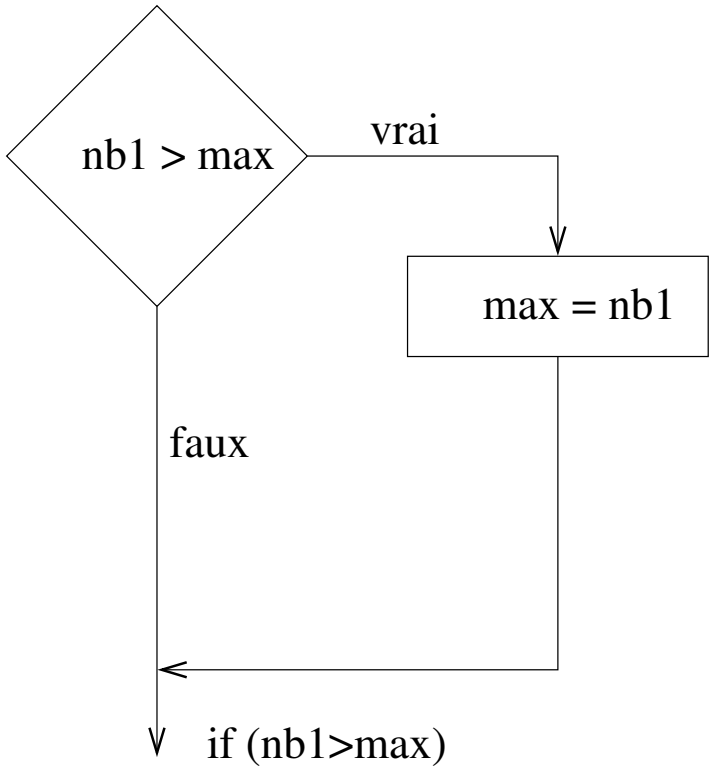
Énoncé `if` : syntaxe

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

Énoncé `if`




```
if (nb1>nb2)
        max=nb1;
else
        max=nb2:
```




```
if (nb1>max)
        max=nb1;
```

Énoncé composé

- Énoncé simple vs énoncé composé

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
  }
  ```
  ```

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`.

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 ?

Lisibilité des `if` emboîtés

- Comment écrire une multitude de `if` emboîtés ?
- Problème de lisibilité accru

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.

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).

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).

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.

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 » ;

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 » ;

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 » ;

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

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 ;
}
```

Structures sélectives

UNIVERSITÉ DE SHERBROOKE

Salaires

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.

Structures sélectives

UNIVERSITÉ DE SHERBROOKE

Présentation des sorties

- On peut améliorer la présentation de la sortie (fixer le nombre de décimales, exiger la notation décimale ou la notation scientifique...)

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>`

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.

Présentation des sorties

- fixed, scientific

Présentation des sorties

- fixed, scientific
- left, right

Présentation des sorties

- fixed, scientific
- left, right
- setw (iomanip)

Présentation des sorties

- fixed, scientific
- left, right
- setw (iomanip)
- setprecision (iomanip)

Présentation des sorties

- fixed, scientific
- left, right
- setw (iomanip)
- setprecision (iomanip)
- setfill (iomanip)

Présentation des sorties

- fixed, scientific
- left, right
- setw (iomanip)
- setprecision (iomanip)
- setfill (iomanip)
- setbase (iomanip)

Présentation des sorties

```
#include <iostream>
#include <iomanip>

double f1=33.1234567899;
int val1 = 34;
cout << setprecision(9) << f1 << endl;
cout << fixed  << f1 << endl;
cout << scientific << f1 << endl;
cout << setw(10) << left <<val1 << endl;
cout << setw(10) << right << val1 << endl;
cout << setw(10) << 77 << endl;
cout << 110 << " = 0x"<< setbase(16)
     << 110 << " = o" << setbase(8) << 110 << endl;
```

Présentation des sorties

```
#include <iostream>
#include <iomanip>

double f1=33.1234567899;
int val1 = 34;

cout << "--------------------------" << endl
     << "Tests pour la precision   " << endl
     << "--------------------------" << endl;
cout << "Defaut (6): "<< f1 << endl  ;
cout << "De 5:       "<< setprecision(5) << f1 << endl;
cout << "Precedante: "<< f1 << endl  ;
cout << "De 9:       "<< setprecision(9) << f1 << endl;
cout << "Precedante: "<< f1 << endl  ;
```

Présentation des sorties

```
--------------------------
Tests pour la precision
--------------------------
Defaut (6):         33.1235
De 5:               33.123
Precedante:         33.123
De 9:               33.1234568
Precedante:         33.1234568
```

Présentation des sorties

```
     << "Tests pour la notation    " << endl
     << "-------------------------" << endl;
cout << "Fixe (9):          "<< fixed  << f1 << endl;
cout << "Scientifique (9): " << scientific << f1 << endl;
cout << "---------------------------------------" << endl
     << "Teste pour la largeur et l'alignement  " << endl
     << "---------------------------------------" << endl;
cout<<"largeur(10) - gauche : "<<setw(10)<<left<<val1<< endl;
cout<<"largeur(10) - droite : "<<setw(10)<<right<<val1<< endl;
cout<<"largeur(10) - droite : "<<setfill('x')
    << setw(10) << 77 << endl;
cout << "largeur (defaut)       : "<< val1 << endl;
```

Présentation des sorties

```
--------------------------
Tests pour la notation
--------------------------
Fixe (9):          33.123456790
Scientifique (9): 3.312345679e+01
---------------------------------------
Tests pour la largeur et l'alignement
---------------------------------------
largeur (10) - gauche : 34
largeur (10) - droite :         34
largeur (10) - droite : xxxxxxxx77
largeur (defaut)      : 34
```

Présentation des sorties

```
cout << "----------------------------------------" << endl
     << "          Tests pour la base            " << endl
     << "----------------------------------------" << endl;
cout << "Base 10, 16 et 8 : " << 110 << " = 0x"<< setbase(16)
     << 110 << " = o" << setbase(8) << 110 << endl;
cout << "Derniere base (110): "<< 110 << endl  ;
cout << "Dernier ajustement : "<< setw (10)<< val1 << endl  ;
```

Présentation des sorties

```
---------------------------------------
          Tests pour la base
---------------------------------------
Base 10, 16 et 8   : 110 = 0x6e = o156
Derniere base (110): 156
Dernier ajustement : xxxxxxxx42
```

Structures sélectives

Jeu d'essais

- Comment vérifier l'exactitude des résultats.

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'implḿentation) 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.

Jeu d'essais

Jeu de tests : Pour être assuré que l'algorithme fonctionne (avant l'implḿentation) 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 :

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.

Jeu d'essais

Jeu de tests : Pour être assuré que l'algorithme fonctionne (avant l'implḿentation) 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.

Structures sélectives

Modifiabilité

- On peut modifier une solution existante (maintenance).

Modifiabilité

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

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.

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.

Structures sélectives

Exemples d'erreurs

- F-18 - pas de «else» dans un «if» car condition supposément jamais fausse.

Exemples d'erreurs

- F-18 - pas de «else» dans un «if» car condition supposément jamais fausse.
- Lancement de fusée raté utilisait nombre de secondes dans une journées = 86400 (plutôt que le temps sidéral - 86,164.1 secondes)

Exemples d'erreurs

- F-18 - pas de «else» dans un «if» car condition supposément jamais fausse.
- Lancement de fusée raté utilisait nombre de secondes dans une journées = 86400 (plutôt que le temps sidéral - 86,164.1 secondes)
- Capsule Mercury - mauvais calcul de la durée d'une orbite. Entrée manuelle dans l'atmosphère (terre au mauvais endroit lors de la rentrée) (n'utilisait pas le temps sidéral).

Exemples d'erreurs

- F-18 - pas de «else» dans un «if» car condition supposément jamais fausse.
- Lancement de fusée raté utilisait nombre de secondes dans une journées = 86400 (plutôt que le temps sidéral - 86,164.1 secondes)
- Capsule Mercury - mauvais calcul de la durée d'une orbite. Entrée manuelle dans l'atmosphère (terre au mauvais endroit lors de la rentrée) (n'utilisait pas le temps sidéral).
- Bateau de guerre américaine - mauvais ciblage
  Fait feu sur un navire marchand mexicain (erreur de 180°).

Exemples d'erreurs

- F-18 - pas de «else» dans un «if» car condition supposément jamais fausse.
- Lancement de fusée raté utilisait nombre de secondes dans une journées = 86400 (plutôt que le temps sidéral - 86,164.1 secondes)
- Capsule Mercury - mauvais calcul de la durée d'une orbite. Entrée manuelle dans l'atmosphère (terre au mauvais endroit lors de la rentrée) (n'utilisait pas le temps sidéral).
- Bateau de guerre américaine - mauvais ciblage
  Fait feu sur un navire marchand mexicain (erreur de 180°).
- Panne des longues distance de AT&T (1990)
  Mauvaise utilisation d'un énoncé break dans un «if» et un «switch».

Structures sélectives

Exercices

Que fait le programme suivant :

```
int main()
{
   float valeur;

   cout << "Entrez la valeur : " ;
   cin  >> valeur ;

   if (valeur < -459.67)
       cout << "Aie! Très froid....." << endl;
   else if (valeur < -40.0 )
       cout << "Hum... C'est du pareil au meme." << endl;
   else if (valeur < 32.0)
       cout << " Crac. Commence a faire froid..." << endl;
   else if (valeur < 212.0)
       cout << "Ok..Ca se réchauffe..." << endl;
   else cout << "Ouf!! Il fait chaud" << endl;

   return 0;
}
```

Exercices

1. Écrivez un programme qui transforme une note de musique de la série constituée par les lettres de l'alphabeten sa note équivalente dans la série constituée de syllabes. N.B. : les notes sont A (la), B (si), C (do), D (ré), E (mi), F (fa) et G (sol).

Exercices

1. Écrivez un programme qui transforme une note de musique de la série constituée par les lettres de l'alphabeten sa note équivalente dans la série constituée de syllabes. N.B. : les notes sont A (la), B (si), C (do), D (ré), E (mi), F (fa) et G (sol).
2. Écrivez un programme qui fait l'inverse de 1.

Exercices

1. Écrivez un programme qui transforme une note de musique de la série constituée par les lettres de l'alphabeten sa note équivalente dans la série constituée de syllabes. N.B. : les notes sont A (la), B (si), C (do), D (ré), E (mi), F (fa) et G (sol).
2. Écrivez un programme qui fait l'inverse de 1.
3. Écrivez un programme qui reçoit une vitesse v en entrée et qui attribue une amende selon les normes suivantes : $v < 50$ (aucune amende), $50 \leq v \leq 70$ (amende de \$100), $70 \leq v < 80$ (amende de \$200), $80 \leq v < 90$ (amende de \$400) et $v \geq 90$ (amende de \$800).