M1102 - Amphi2 C++

Alain Casali Marc Laporte

Aix Marseille Univ

Identificateur

`char`

Valeurs

N'importe quel caractère ASCII entre ‘
ACSII : American Standard Code for Information Interchange

Opérations de comparaison

Produisent un booléen

< <= > >=

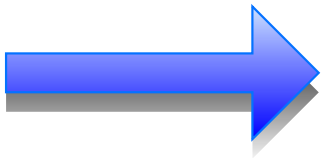
Opérations d'identité

Produisent un booléen

== !=

Élément suivant

Produit un `car`
La fonctions `succ ()` n'existe pas en C++.



Pour passer au caractère suivant, on applique l'opérateur `++` à gauche de la variable.

Exemple

```
char C = 'a';
char Next = ++C;
cout << Next;
```

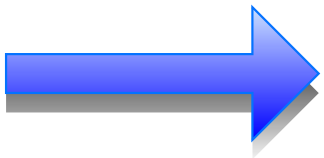
b

```
char Next = ++'a';
cout << Next;
```

Erreur de compilation

Élément précédant

Produit un `car`
La fonctions `pred ()` n'existe pas en C++.



Pour passer au caractère suivant, on applique l'opérateur `--` à gauche de la variable.

Exemple

```
char C = 'b';
char Prev = --C;
cout << Prev;                    a
```

Fonctions applicables

Produisent un `car`

Pré-requis pour utiliser les fonctions sur les caractères :

```
#include <cctype>
```

- `tolower (UnCaract)`

  renvoie le caractère en minuscule (uniquement si c'est une majuscule)

- `toupper (UnCaract)`

  renvoie le caractère en majuscule (uniquement si c'est une minuscule)

Exemple

```
cout << toupper (Next);        B
cout << tolower ('C');         c
```

Prédicats applicables

Produisent un `bool`

Pré-requis pour utiliser les prédicats sur les caractères :

```
#include <cctype>
```

- `islower (UnCaract)`

  renvoie vrai si le caractère est une minuscule

- `isalpha (UnCaract)`

  renvoie vrai si le caractère est alphanumérique

D'autres prédicats sont disponibles sur : `http://www.cplusplus.com/reference/cctype/?kw=cctype`

Exemple

```
if (islower (Next))
{
    cout << "caractere minuscule";
}
```

Caractères spéciaux

Liste (non exhaustive) des caractères spéciaux :

Caractère	Signification
`'\t'`	tabulation
`'\n'`	Line Feed (saut de ligne)
`'\r'`	Carriage Return (retour chariot)
`'\' '`	caractère '

Dans les systèmes compatibles Unix, le caractère matérialisant un retour à la ligne dans un fichier est `'\n'`, alors que dans les systèmes compatibles Windows c'est la combinaison des deux caractères `'\r' '\n'`

Encodage des caractères

Le type `char` est codé sur 1 octet $\longrightarrow$ 256 valeurs possibles.
Le type `wchar_t` est codé sur 4 octets $\longrightarrow$ 2^{32} valeurs possibles.
Encodages populaires en France :

- Unicode (compatible avec UTF-8, UTF-16 et UTF-32) ;
- ISO 8859-15.

NB : UTF signifie Universal Transformation Format

Identificateur

`string`

Valeurs

N'importe quelle suite de caractères ASCII entre double quotes "

Opérations de comparaison

Produisent un booléen

`<` `<=` `>` `>=`

$\Longrightarrow$ il existe une relation d'ordre (ordre lexicographique)

Opérations d'identité

Produisent un booléen

`==` `!=`

Taille d'une string

Produit un `size_t` ⇔ `unsigned long`
Pour connaître la taille d'une `string`, on appelle la méthode `size ()` sur l'objet (la variable) de type `string`.

Exemple

```
string Str;
cin >> Str;                    Mario
cout << Str.size();            5
```

Accès à un élément

Produit un `char`
On accède au $(i+1)^{ème}$ élément d'une `string` en utilisant `[i]`.

- La première case de la string a pour indice 0 ;
- La dernière case de la string a pour indice `VarIdent.size () - 1` ;
- Aucun contrôle quant à la validité de l'indice n'est effectué lors d'un accès à une case, que ce soit en lecture ou en écriture.

Exemple

Afficher le contenu d'une `string` en allant à la ligne après chaque caractère

```
pour (i variant_de 0 a Str.size() - 1)
{
    cout << Str [i] << endl;
}
```

Opérateur de concaténation +

Produit un `string`

L'opérateur de concaténation + est un opérateur binaire. Pour qu'il puisse produire une `string`, on doit avoir :

- À gauche un littéral ou une variable (lhs) ;
- À droite un littéral ou une variable (rhs).

lhs	rhs	Compatibilité
`char`	`string`	C++11
`string`	`char`	C++11
`string`	`string`	C++

Exemple

```
Str = "0123456789";
cout << Str.size ();          10
Str = Str + "aeiouy\"";
cout << Str.size ();          17
```

Boucle while

Modèle général algorithmique :

```
tant_que (condition)
faire
    ...
ffaire
```

Modèle général C++ :

```
while (condition)
{
    ...
}
```

Pas de ';' après l'instruction while (condition).

Exemple

Algo :

```
declarer i : entier_naturel;
i <- 0;
tant_que (i < taille (Str))
faire
    afficher (Str[i]);
    ligne_suivante;
    i <- i + 1;
ffaire
```

C++ :

```
unsigned i;
i = 0;
while (i < Str.size())
{
    cout << Str [i]
         << endl;
    i = i + 1;
}
```

Modèle général algorithmique :

```
repeter

    ...

tant_que (condition)
```

Modèle général C++ :

```
do
{
    ...
}
while (condition);
```

Le ';' est obligatoire après l'instruction `while (condition)`.

Exemple

```
unsigned i;
i = 0;
do
{
    cout << Str[i] << endl;
    i = i + 1;
}
while (i < Str.size ());
```

Le code est faux si `Str` est une `string` de taille nulle !

Boucle for

Modèle général algorithmique :

```
pour (NomVar variant_de BorneMin a BorneMax)
faire
    ...
ffaire
```

Cette boucle n'existe pas telle qu'elle en C++, mais la boucle `for` existe.

Profil de la boucle for

Modèle général C++ :

```
for ([expression1]; [expression2]; [expression3])
{
    instruction(s);
}
```

- `instruction(s)` : un ensemble d'instructions
- `expression1` : une instruction unique, exécutée avant l'entrée dans l'itération
- `expression2` : une instruction unique et booléenne représentant la condition de continuation
- `expression3` : une instruction unique, exécutée avant la prochaine itération

Pas de ';' après l'instruction `for ( ; ;)`.

Exemple

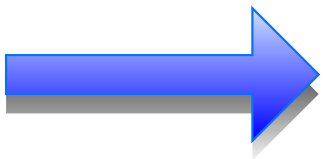
```
for (unsigned i = 0; i < Str.size (); i = i + 1)
{
    cout << Str[i] << endl;
}
```

Équivalent à :

```
for (unsigned i = 0; i < Str.size (); )
{
    cout << Str[i] << endl;
    i = i + 1;
}
```

Équivalent à : ?

```
unsigned i = 0;
for (; i < Str.size (); i = i + 1)
{
    cout << Str[i] << endl;
}
```



Non : portée de variable

Boucle infinie

Modèle général algorithmique :

```
boucle
    instruction(s);
fboucle
```

Modèle général C++ :

```
for ( ; ; )
{
    instruction(s);
}
```

```
while (true)
{
    instruction(s);
}
```

Pas de condition de sortie explicite.

1. Le type caractère
2. Le type `string`
3. Boucle `while`
4. Boucle `for`
5. Boucle infinie
6. Rupture de schéma répétitif
   - instruction `break`
   - instruction `continue`
7. Schéma séquentiel `switch / case`
8. Les sous-programmes
9. Cosmétique
10. Le type `vector` (tableau de taille variable)

Rupture de schéma répétitif : instruction `break`

Modèle général algorithmique :

```
boucle
    instruction (s);
    si (condition) sortie;
    instruction (s);
fboucle
```

Modèle général C++ :

```
for([e1]; [e2]; [e3])
{
    instruction(s);
    if (condition) break;
    instruction(s);
}
```

Exemple 1 : générer un entier naturel > 10

Exemple

```
declarer N : entier_naturel;
boucle
    afficher ("Entrer une valeur > 10 : ");
    saisir (N);
    si (N > 10) sortie;
    afficher (">10 svp!");
fboucle
```

Exemple

```
unsigned N;
while (true)
{
    cout << "Entrer une valeur > 10 : ";
    cin >> N;
    if (N > 10) break;
    cout << ">10 svp!";
}
```

Rupture de schéma répétitif : instruction `continue`

Modèle général algorithmique :

```
boucle
    instruction (s);
    si (condition) continue;
    instruction (s);
fboucle
```

Modèle général C++ :

```
for([e1]; [e2]; [e3])
{
    instruction(s);
    if (condition) contnue;
    instruction(s);
}
```

Exemple : afficher une lettre sur deux d'un mot saisi

Exemple

```
string Str;
cin >> Str;
for (unsigned i = 0; i < Str.size (); i = i + 1)
{
    if (0 == i%2)
    {
        cout << Str [i];
    }
}
```

2 blocs ⇒ 2 indentations
⇒ 2 fois moins lisible

Exemple

```
string Str;
cin >> Str;
for (unsigned i = 0; i < Str.size (); i = i + 1)
{
    if (0 != i%2) continue;
    cout << Str [i];
}
```

1 blocs ⇒ 1 indentations
⇒ plus lisible

Combinaison de break et de continue

Exemple

```
while (ICanFrag ())
{
    if (Kill ()) continue;
    if (SomeoneNearMe ()) break;
}
FindAnotherSpot ();
```

1. Le type caractère
2. Le type `string`
3. Boucle `while`
4. Boucle `for`
5. Boucle infinie
6. Rupture de schéma répétitif
7. Schéma séquentiel `switch / case`
   - Absence d'instruction `break` dans un `switch/ case`
   - Instruction `break` et boucle
8. Les sous-programmes
9. Cosmétique
10. Le type `vector` (tableau de taille variable)

Schéma séquentiel switch / case

Modèle général algorithmique :

```
choix_sur Variable_de_choix
entre
  cas Ens_1 :
    Sequ_1;
  cas Ens_2 :
    Sequ_2;
  cas Ens_3 :
    Sequ_3;
  autre :
    Sequ_4;
fchoix
```

Modèle général C++ :

```
switch (VarIdent)
{
case Lit1:
     Instr1;
     break;
case Lit2:
     Instr2;
     break;
...
default:
     Instr4;
}
```

L'instruction `break` permet la sortie du bloc courant (i.e. la sortie du `switch` / `case`)

- `VarIdent` doit être de type entier ou caractère ;
- Pas de `break` après `Instr4` ;

Exemple : gérer le déplacement (touches uniques)

Exemple

```
char c;
cin >> c;
switch (c)
{
case 'a':
     MouveLeftward ();
     break;
case 'z':
     MouveUpward ();
     break;
case 'e':
     MouveRightward ();
     break;
case 's':
     MouveDownward ();
//   break;
}
```

Remarque : pas de {} pour les blocs case.

Exemple : gérer le déplacement (touches multiples - 1)

Exemple

```
char c;
cin >> c;
switch (c)
{
case 'A':
case 'a':
    MouveLeftward ();
    break;
case 'Z':
case 'z':
    MouveUpward ();
    break;
case 'E':
case 'e':
    MouveRightward ();
    break;
case 'S':
case 's':
    MouveDownward ();
}
```

Exemple : gérer le déplacement (touches multiples - 2)

Exemple

```
char c;
cin >> c;
switch (tolower (c))
{
case 'a':
     MouveLeftward ();
     break;
case 'z':
     MouveUpward ();
     break;
case 'e':
     MouveRightward ();
     break;
case 's':
     MouveDownward ();
}
```

Exemple : gérer le déplacement (touches multiples - 2)

Exemple

```
char c;
cin >> c;                          a
switch (tolower (c))
{
case 'a':
     MouveLeftward ();             Deplacement gauche
//   break;
case 'z':
     MouveUpward ();               Deplacement haut
     break;
case 'e':
     MouveRightward ();
     break;
case 's':
     MouveDownward ();
}
```

Puisqu'il n'y a pas d'instruction `break`, le `case` est ignoré.

Instruction break et boucle (1)

Exemple

```
char c;
cin >> c;
for ( ; 'q' != tolower (c); cin >> c)
{
    switch (tolower (c))
    {
    case 'a':
        MouveLeftward ();
        break;
    case 'z':
        MouveUpward ();
        break;
    case 'e':
        MouveRightward ();
        break;
    case 's':
        MouveDownward ();
    }
...
}
```

Instruction break et boucle (2)

Pas du tout équivalent à :

Exemple

```
char c;
cin >> c;
for ( ; 'q' != tolower (c); cin >> c)
{
    if ('a' == tolower (c))
    {
        MouveLeftward ();
        break;
    }
    else if ('z' == tolower (c))
    {
        MouveUpward ();
        break;
    }
    ...
}
```

Dans une boucle, l'instruction `break` provoque la sortie de la boucle (la plus interne).

1. Le type caractère
2. Le type `string`
3. Boucle `while`
4. Boucle `for`
5. Boucle infinie
6. Rupture de schéma répétitif
7. Schéma séquentiel `switch / case`
8. Les sous-programmes
   - Procédure
   - Fonction
   - Passage de paramètres donnés
   - Passage de paramètres [données] résultats
9. Cosmétique
10. Le type `vector` (tableau de taille variable)

Procédure

Modèle général algorithmique :

```
procedure NomProc (Variable1 : [in|out|in_out] Type1,
                   Variable2 : [in|out|in_out] Type2, ...);
```

Modèle général C++ :

```
void ProcName (VarIdent1 : [in|out|in_out] Type1,
               VarIdent2 : [in|out|in_out] Type2, ...);
```

Exemple

Procédure qui affiche "bonjour" En algorithmique :

```
procedure AffichBonjour ()
debut
    afficher ("bonjour");
    ligne_suivante;
fin
```

En C++ :

```
void ShowHello ()
{
    cout << "bonjour" << endl;

}// ShowHello ()
```

Fonction

Modèle général algorithmique :

```
fonction NomFonct (Variable1 : [in|out|in_out] Type1,
                   Variable2 : [in|out|in_out] Type2, ...)
renvoie TypeDeRetour;
```

Modèle général C++ :

```
ReturnType FctName (VarIdent1 : [in|out|in_out] Type1,
                    VarIdent2 : [in|out|in_out] Type2,...);
```

A l'intérieur d'une fonction C++, l'instruction algorithmique `renvoie` se traduit par `return`.

Exemple

Fonction qui affiche "bonjour" et renvoie cette `string`

En algorithmique :

```
fonction AffichEtRenvoieBonjour () renvoie string
debut
    afficher ("bonjour");
    ligne_suivante;
    renvoie "bonjour";
fin
```

En C++ :

```
string ShowHello ()
{
    cout << "bonjour" << endl;
    return "bonjour";
} // ShowHello ()
```

Passage de paramètres données

Modèle général algorithmique :

```
SP NomSP (Variable1 : in Type1, Variable2 : in Type2, ... )
[renvoie TypeDeRetour];
```

Modèle général C++ :

```
[ReturnType] FctName (const Type1 & VarIdent1,
                      const Type2 & VarIdent2, ...);
```

Exemple : renvoyer le nombre de caractères d'espacement d'une `string` passée en paramètre

Exemple

```
unsigned SpaceCount (const string & Str)
{
    unsigned Cpt = 0;
    for (unsigned i = 0; i < Str.size (); i = i + 1)
    {
        if (isspace (Str[i])) Cpt = Cpt + 1;
    }
    return Cpt;
} //SpaceCount ()
```

Passage de paramètres [donnés] résultats

Modèle général algorithmique :

```
SP NomSP (Variable1 : [in_]out Type1, Variable2 : [in_]out Type2, ... )
[renvoie TypeDeRetour];
```

Modèle général C++ :

```
[ReturnType] FctName (Type1 & VarIdent1, Type2 & VarIdent2, ...);
```

Exemple : écrire la procédure qui calcule un entier aléatoire entre les bornes Min et Max

Exemple

```
void Rand (const int & Min, const int & Max, int & Res)
{
    srand (time (NULL));
    Res = Min + rand () % (Max - Min);
} // Rand ()
```

1. Le type caractère
2. Le type `string`
3. Boucle `while`
4. Boucle `for`
5. Boucle infinie
6. Rupture de schéma répétitif
7. Schéma séquentiel `switch / case`
8. Les sous-programmes
9. Cosmétique
   - Manipulateur `setw ()` ou cadrage à droite
   - Manipulateur `setfill ()`
10. Le type `vector` (tableau de taille variable)

Manipulateur setw () ou cadrage à droite

Pré-requis :

```
#include <iomanip>
```

Profil :

```
/*undefined*/ setw (int n);
```

Définition :

L'injection, dans un flux de sortie (cout), du manipulateur setw (Nb), provoque l'affichage de la variable ou du littéral qui le suit sur Nb caractères (cadrés à droite). Les caractères manquants sont remplacés par des espaces.

Exemple

```
int a (10), b (20), c (30), d (40);
cout << setw (4) << a << setw (4) << b   __10__20
        << endl;
cout << setw (4) << c << setw (4) << d   __30__40
        << endl;
```

Manipulateur setfill ()

Pré-requis :

```
#include <iomanip>
```

Profil :

```
/*undefined*/ setfill (char c);
```

Définition :
Le manipulateur `setfill ()` s'utilise avant `setw ()`. Il provoque le remplacement des espaces par le caractère passé en paramètre. `setfill ()` a un effet permanent.

Exemple

```
int a (10), b (20), c (30), d (40);
cout << setfill ('x') << setw (4) << a
     << setfill ('?') << setw (4) << b  << endl      xx10??20
     << setw (4) << c
     << setfill ('␣') << setw (4) << d
     << endl;                                        ??30__40
```

Pré-requis

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

Identificateur

Modèle général algorithmique :

```
tableau_de UnType;
```

Modèle général C++ :

```
vector <Type>
```

Exemple

```
vector <int> VInt;
vector <float> VFloat;
```

La taille du vecteur est nulle lors de sa création

Taille d'un vecteur

Pour connaître la taille d'un `vector ()`, on appelle la `méthode () size ()` size () sur l'objet (la variable) de type `vector` (même nom que pour connaître la taille d'une `string ()`).

Exemple

```
cout << VInt.size ();   // 0
```

Insertion en fin de vecteur

Pour insérer des éléments compatibles en fin d'un `vector`, on appelle la `méthode () push_back ()` sur l'objet (la variable) de type `vector` (cette méthode existe aussi pour les `string`). Le vecteur est redimensionné automatiquement.

Exemple

```
VInt.push_back (0);
VInt.push_back (1);
...
VInt.push_back (9);
cout << VInt.size ();   //10
```

- Même exemple que précédemment en utilisant une boucle `for` :

Exemple

```
for (int InsVal (0); InsVal < 10; InsVal = InsVal + 1)
{
    VInt.push_back (InsVal);
}
cout << VInt.size ();  //10
```

- Insertion de 10 valeurs saisies au clavier :

Exemple

```
int InsVal;
for (unsigned NbVal = 0; NbVal < 10; NbVal = NbVal + 1)
{
    cin >> InsVal;
    VInt.push_back (InsVal);
}
cout << VInt.size ();  //10
```

Redimensionnement d'un vecteur

Pour redimensionner un `vector`, on appelle la `méthode () resize () sur l'objet` (la variable) de type `vector`.

Exemple

```
VInt.resize (42);
```

Si la taille de l'ancien vecteur $<$ taille du nouveau (i.e. on ne fait pas de troncature), alors les nouvelles valeurs sont :

Type	Valeur par défaut
Entier	0
Réel	0.0
`string`	chaine vide
`char`	'\0'

Accès à un élément en lecture / écriture

On accède au $(i+1)^{ème}$ élément d'un `vector` en utilisant la notation `[i]`.

- La première case de lu `vector` a pour indice 0 ;
- La dernière case du `vector` a pour indice `VarIdent.size () - 1` ;
- Aucun contrôle quant à la validité de l'indice n'est effectué lors d'un accès à une case, que ce soit en lecture ou en écriture.

Exemple

Affichage du contenu d'un vecteur

```
for (unsigned i = 0; i < VInt.size (); i = i + 1)
{
    cout << VInt [i];
}
```

Et surtout pas `cout « VInt;`

La boulette du débutant

Exemple

```
vector <int> VInt;
VInt [0] = 0;
```

Compilation : pas d'erreur
Exécution : Segmentation fault : 11 (anciennement écran bleu)

A l'aide

`VInt` est déclaré, mais non dimensionné

$\Rightarrow$ La case d'indice 0 n'existe pas!!

Recopie d'un tableau dans un autre

On souhaite recopier le contenu de `VInt` dans un second tableau (`VInt2`)

```
vector <int> VInt2;
```

- Solution 1 :

```
for (unsigned i = 0; i < VInt.size (); i = i + 1)
{
    VInt2.push_back (VInt[i]);
}
```

- Solution 2 :

```
vector <int> VInt2;
VInt2 = VInt;
// vector <int> VInt2 = VInt;
```

- Solution 3 :

```
vector <int> VInt2 (VInt);
```