

Principales instructions du langage C / C++

(traduction du document « Notation algorithmique de base »)

1. Structure générale d'un programme

```
#include <iostream>
using namespace std;
int main ( )
{
    // Bloc de déclaration des variables
    Variables

    // Programme principal
    Instructions
}
```

2. Déclaration des variables

```
type_1 nom_var_1 ;
type_2 nom_var_2, nom_var_3 ;
...
const type_1 nom_const_1 = valeur_constante;
```

3. Types

int (**entier**), float (**réel**), bool (**booléen**), char (**caractère**), char [taille](**chaîne de caractères**)

4. Commentaires

```
// Exemple de commentaire
```

5. Séquence

```
{
    action_1
    action_2
    ...
}
```

6. Action de lecture/écriture

```
cin >> var_1 >> var_2 >> ... >> var_n ;
cout << expression_1 << expression_2 << .... << expression_n ;
Pour le retour à la ligne utiliser endl à la place de expression_i dans la ligne cout
```

7. Affectation

```
variable = expression ;
```

8. Instruction conditionnelle

```
if (condition)
    action_si_condition_vérifiée
else
    action_sinon
```

9. Choix multiple

```
switch (variable)
{
    case valeur_1 : action_1
                    break ;
    case valeur_2 : action_2
                    break ;
    ...
    default : action_sinon
}
```

10. Actions itératives

```
for (variable = indice_début ; variable <= indice_fin ; variable++)
    action
```

```
while (condition)
    action

do
    action
while (condition) ;
```

11. Sous-programmes

```
<Type_de_retour> nom_fonction(<Paramètres>)
{
    Instructions
}

<Type_de_retour> ::= int | float | ...
<Paramètres> ::= <Type> <Type_paramètre>Nom_paramètre
<Type> ::= int | float | ...
<Type_paramètre> ::= Ø | &
```

12. Fichiers

Utilisation d'une variable pointeur_fichier de type FILE *

Ouverture : pointeur_fichier = fopen(nom_fichier,<mode>) ;
 <mode> ::= "r" | "w" | "a"

Fermeture : fclose(pointeur_fichier) ;

Lecture / écriture binaire :

```
fread(données, taille_une_donnée, nombre_données, pointeur_fichier) ;
fwrite(données, taille_une_donnée, nombre_données, pointeur_fichier) ;
```

Lecture / écriture ASCII :

```
fscanf(pointeur_fichier, format, données) ;
fprintf(pointeur_fichier, format, données) ;
```

Fin fichier : feof(pointeur_fichier) (renvoie 0 si la fin de fichier n'est pas atteinte)

Décalage : fseek(pointeur_fichier, decalage, <position>) ;
 <position> ::= SEEK_SET | SEEK_END | SEEK_CUR

13. Structures

```
struct nom_structure
{
    Définition des éléments...
} ;
```

Accès à un élément de la structure : variable.nom_élément

14. Vecteurs et tableaux

```
type_éléments nom_tableau [dim_1][dim_2][...] ;
```

15. Pointeurs

```
type_pointé *nom_pointeur ;
```

Opérateur Adresse : &nom_variable

Accès à l'élément pointé : *nom_pointeur

Pour accéder à un champ d'un pointeur sur enregistrement, les notations
 nom_pointeur->champ et (*nom_pointeur).champ sont équivalentes

Allocation mémoire : nom_pointeur = new type_pointé ;

Désallocation mémoire : delete nom_pointeur ;