

Éléments de base

- Programmation ? C'est quoi ça ?
- Les caractéristiques du C
- Éléments de base en C
- Les données
- Les variables
- Règles d'évaluation

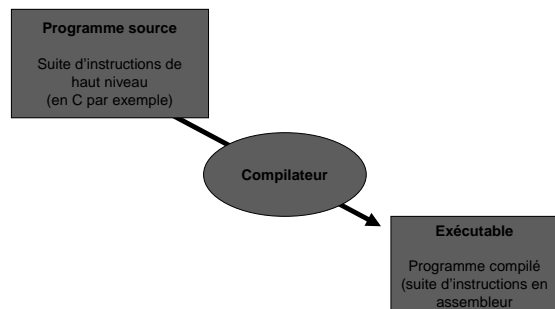
Qu'est ce que la programmation ?

- L'art de transcrire un problème en une suite d'instructions compréhensibles par l'ordinateur

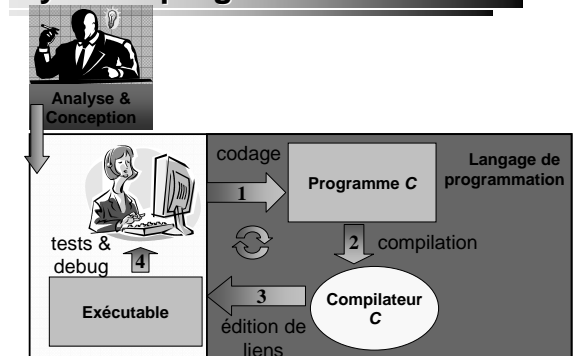
Les niveaux de programmation

- Assembleur
 - Langage constitué du set d'instructions d'un processeur (bas niveau)
 - Particulier à un type de processeur
- Compilation
 - Transformation d'un langage de haut niveau en assembleur
 - Plus facile d'accès
 - Indépendant du processeur

Compilation



Cycle de programmation



Caractéristiques du C

- Langage structuré
 - ⇒ Organisation en blocs
- Langage déclaratif
 - ⇒ Les éléments doivent être déclarés
- Format libre
 - ⇒ Liberté de mise en page
- Langage modulaire
 - ⇒ Regroupement de fonctionnalités par module
- Langage compilé
 - ⇒ Modules peuvent être compilés séparément

Terminologie

- Code source et « header » (.c .cpp .h)
- Objet compilé (.obj .o)
- Librairie, bibliothèque de fonction (.lib .a)
- Librairie partagée/dynamique (.dll .so)
- Exécutable (.exe .x)

Point d'entrée d'un programme C

- Fonction **main()**
 - Contient les instructions du programme
 - Point d'entrée = 1^{ère} fonction exécutée

Éléments de base du C

- Commentaires
- Instruction élémentaire
- Bloc de contrôle
- Nombres
- Caractères
- Chaîne de caractère
- macro

Commentaires

// Voici des commentaires sur ligne

/* Voici des commentaires
Sur plusieurs lignes. On met normalement
Des commentaires dans tous
les programmes écrits en C ou non */

Instruction élémentaire, bloc

Syntaxe

Instruction
commande;

Bloc

```
{
  instruction 1;
  instruction 2;
  ....
}
```

Exemple

```
cout << "Hello World" << endl;
```

```
{
  cout << "Hello" << endl;
  {
    cout << endl;
  }
  cout << "Good Bye" << endl;
}
```

Exemple de bloc avec commentaires

```
/* Et voilà un bloc qui écrit deux phrases sur la même
console. Il est toujours préférable de mettre qu'une seule
instruction par ligne */
{
    cout << "Hello world!" << endl; // premiere instruction
    cout << "GoodBye" << endl;
    // fin du programme
}
```

Une bonne présentation du code = simplification pour trouver des erreurs !
UTILISER DES TABULATIONS !!

Un programme simple

```
#include <iostream> // inclure la librairie E/S
using namespace std;
// nécessaire pour toutes les librairies STL

/* Programme qui affiche "Hello, cruel world !" sur
la console. */
int main()
{
    cout << "Hello, cruel world !" << endl;
}
```

Les données : les nombres

- Nombres entiers :
 - Suite de chiffres, peuvent être précédés d'un signe + ou -, par exemple :
 - 42 +327 -13678
- Nombres réels :
 - Comportent un point, peuvent être précédés d'un signe + ou -, par exemple
 - +3.14 6.28 -0.3456
- Notation scientifique:
 - 6.23 10⁷ s'écrit 6.23E+7
 - 4.24 10⁻¹⁵ s'écrit 4.24E-15

Les données : les caractères

- Encadré d'apostrophes
 - Par exemple : 'a' 'T' '+' '1' 'f'
- 3 catégories de caractères utilisables:
 - Les lettres 'A'-'Z' et 'a'-'z'
 - Les chiffres '0'-'9'
 - Les caractères spéciaux '+' '=' ';' ':' '.'
- Autres caractères spéciaux
 - '\n' nouvelle ligne '\t' tabulation
 - '\\' backslash '\\' apostrophe
 - '\"' guillemet

Les données : les chaînes de caractères (constantes)

Encadrée de symboles ""

Par exemple, cette instruction:

```
cout << "il est grand" << endl;
```

Affiche : il est grand

La macro directive #define

- Associe un nom à des valeurs en les déclarant avant leur utilisation


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

#define PI 3.14159
#define RADIUS 12
#define TEXT "Contenu du cercle : "

void main()
{
    cout << TEXT << PI*RADIUS*RADIUS
    << endl;
}

Résultat :
Contenu du cercle : 452.38896
```

La notion de variable

Associe un nom à une valeur numérique

Doit commencer par une lettre

Déclarée avant son utilisation

Déclaration d'une variable:
Identificateur de **type** suivi du **nom**

4 identificateurs standards

int variables entières
16 bits \Rightarrow -32767 à 32768

float variables réelles (ou flottantes) en simple précision
32 bits \Rightarrow env. 6 digits ($3.4E+/-38$)

double variables réelles (ou flottantes) en double précision
64 bits \Rightarrow env. 12 digits ($1.7E+/-308$)

char variables de type caractère
8 bits \Rightarrow 255 caractères

Exemples de déclarations

```
int x2;           float abcABC;
char c;
int 2x;         double ADoubleV,
                  coursLCI;

char variableOne;
char x, test2;


int x,y,z;        int x;
                  int y;
                  int z;
```

Préfixes *short*, *long*, *signed* et *unsigned*

int	16 bits	-32767 à 32768			
unsigned int	16 bits	0 à 65535	unsigned long int	32 bits	0 à 4294967296
short int	8 bits	-128 à 127	float	32 bits	env. 6 digits précision
unsigned short int	8 bits	0 à 255	double	64 bits	env. 12 digits précision
long int	32 bits	-2147483648 à 2147483649	long double	128 bits	env. 24 digits précision

Notation abrégées

- short** pour **short int**
- long** pour **long int**
- unsigned** pour **unsigned int**

- Exemples
- short int** a, b;
 - unsigned int** somme;
 - long double** c;
- 
- short** a, b;
 - unsigned** somme;
 - long double** c;

Affectation

Syntaxe

variable = expression;

Stocke dans la variable la valeur de l'expression

Exemple

int x; // déclaration de x

x = 3; // donne la valeur 3 à x

Déclaration et affectation combinées

en une seule expression :

```
// declaration de la variable x et initialisation avec la valeur 3
int x = 3;

char variableOne = 'a';
float abcABC;
abcABC = -3.5675f;
double aDoubleVariable = 5.062;
double coursLCI = 5.0;
```

Affichage d'une variable

```
int i;
i = 45;
cout<<"La valeur de la variable i est " << i << endl;
i = 2;
cout<<"La valeur de la variable i est " << i << endl;
```

Exécution :

```
La valeur de la variable i est 45
La valeur de la variable i est 2
```

⇒ printf("La valeur de la variable i est %d\n", i); // version « pur C »

Un type de variable spéciale : string

Type « chaîne de caractères » en C++

Pour l'utiliser il faut introduire une librairie *string*.

Exemple:

```
#include <string>
using namespace std;
int main() {
    // declaration de la string s
    string s;
}
```

string : affectation

Similaire aux types de base (int, char, float, double) :

```
// affectation de la string s
s = "Hello my world !";
// declaration et affectation de s
string s = "Help me!";
```

```
char c = s[0]; // s[0] = caractère H
char d = s[3]; // s[3] = caractère p
// affiche la string sur la console
cout << s << endl;
```

Les expressions mathématiques

- Expression
 - ⇒ combinaison de valeurs numériques (constantes, invocation de fonctions ...), de signes opératoires et de parenthèses
- 4 ou 5 opérateurs sont disponibles
 - l'addition + la soustraction -
 - la multiplication * la division /
 - et le reste de la division **entière** % (modulo)

Exemple

```
int i=20, j=40, k=12;
//déclaration + affectation des variables
i = j + k; // maintenant i a la valeur 52
j = j - 20; // j a la valeur 20
i = i*2 + j; // i a la valeur 124
int d = 40/k; // d a la valeur 3
int r = 40%k; // r a la valeur 4
```

Abréviations

```

j = j + 4; ⇨ j += 4;
j = j - 3; ⇨ j -= 3;
j = j * 2; ⇨ j *= 2;
j = j / 5; ⇨ j /= 5;
j = j % 6; ⇨ j %= 6;

x *= y + 2; ⇨ x = x * (y + 2);
et pas x = x * y + 2; ☹

```

Incrémenter & décrémenter

- Deux opérateurs unaires :
 - `++` additionne 1 à l'opérande
 - `--` soustrait 1 à l'opérande
$$j = j + 1 \Rightarrow j++;$$
- Peuvent être utilisés comme préfixes `++j` ou suffixes `j++`
 - Si $j = 5 \Rightarrow x = j++ \Rightarrow x = 5$ et $j = 6$
 - $x = ++j \Rightarrow x = j = 6$

Autres opérations mathématiques

Disponibles en utilisant des bibliothèques mathématique (math)

```

#include <math.h>
int main() {
    double x = sqrt(100); // racine carrée
    double y = x * sin(0.4) + cos(-0.4);
}

```

Principales règles d'évaluation

- Sans parenthèses `*`, `/`, `%` sont traitées avant `+` et `-`
- A priorité égale, l'évaluation se fait de gauche à droite
- Expressions entre parenthèses sont effectuées en premier
 - Règle variable pour les arguments de fonctions qui sont évalués en premier

Exemple d'ordre d'évaluations

```

3 + 4 * 7 / 2 - 5
⇨ 3 + 28 / 2 - 5
⇨ 3 + 14 - 5
⇨ 17 - 5
⇨ 12

```

```

3 + 4 * 7 / (2 - 5)
⇨ 3 + 4 * 7 / (-3)
⇨ 3 + 28 / (-3)
⇨ 3 + (-9)
⇨ -6

```

Résumé du cours

- Généralités
 - Programmation
 - Compilation
 - Caractéristiques du C
- Pt d'entrée d'1 prog. C
- Commentaires
- Instruction élémentaire, bloc
- Constantes
 - Nombre, caractères, chaînes de caractères
- Variable
 - Affectation & déclaration
- Types
 - Int, char, float, double, string
- Expressions arithmétiques
 - Règles d'évaluation