

LIFAP1 : ALGORITHMIQUE
ET PROGRAMMATION IMPÉRATIVE,
INITIATION

COURS 2 : BASES DU LANGAGE C

OBJECTIFS DE LA SÉANCE

Apprendre la syntaxe du langage C

Savoir traduire un algorithme en langage C

Vous permettre de pouvoir débuter les séances de travaux pratiques

- Environnement C5 / (code::blocks)
- OS : windows (linux)

PLAN

Historique du C

Un programme C

Les entrées / sorties en C

Types des données algorithmique / C

Éléments syntaxiques du langage C

- Structures de contrôles

Traduction d'algorithmes simples en langage C

UN PEU D'HISTOIRE

1945, les programmes étaient écrits directement en code machine...

1954 : FORTRAN 1

1978 : “The C programming language”

- Brian W. Kernighan et Dennis M. Ritchie

1983-1988 : Normalisation ANSI (avec C++)

1988 : “The C programming language : 2ème édition”

- des mêmes auteurs

Évolutions permanentes

PLAN

Historique du C

Un programme C

Les entrées / sorties en C

Types des données algorithmique / C

Éléments syntaxiques du langage C

- Structures de contrôles

Traduction d'algorithmes simples en langage C

PREMIER EXEMPLE EN C

premierexemple.cpp

```
1 #include <iostream.h>
2 #include <iostream>
3 using namespace std;
4
5 int main (void)
6 {
7     cout << "Hello world" << endl ;
8     return 0 ;
9 }
```

Mots clé du langage :

- Identifiés par une couleur particulière dans l'interface C5 et Code::Blocks

Anatomie d'un programme C :

- Nom de fichier d'**extension .cpp**
- Utilisation de bibliothèques (déclarations des fonctions externes) : directive **#include**
 - Pour entrées / sorties
 - Pour opérations mathématiques
 - ...
- Définitions des **sous-programmes** (fonctions et procédures) → CM 3
- Définition de la fonction principale **main**

PREMIER EXEMPLE EN C

premierexemple.cpp

```
1 #include <iostream.h>
2 #include <iostream>
3 using namespace std;
4
5 int main (void)
6 {
7     cout << "Hello world" << endl ;
8     return 0 ;
9 }
```

Fonction principale : main

- Instruction particulière
exit ou return (EXIT_SUCCESS)
- indique si le programme s'est déroulé et terminé normalement

Délimitation des blocs par { et }

Toutes les instructions se terminent par un ";"

cout : permet d'afficher un message à l'écran (c++) (cout <<)

- "..." : chaîne de caractères
- On peut avoir plusieurs <<
- endl : constante C++
permet de passer à la ligne suivante (end-line) après avoir écrit le message

UN PROGRAMME C

Suites ordonnées de déclarations ou de définitions

- de types,
- de variables,
- de sous-programmes (CM 3)

Une fonction particulière : main

- première fonction appelée lors de l'exécution
- appelle les autres sous-programmes

Tout nom doit être déclaré avant d'être utilisé

- type, variable, fonction

Préprocesseur : #include

- directive de compilation pour l'inclusion des déclarations des sous-programmes prédéfinis dans les bibliothèques

PLAN

Historique du C

Un programme C

Les entrées / sorties en C

Types des données algorithmique / C

Éléments syntaxiques du langage C

- Structures de contrôles

Traduction d'algorithmes simples en langage C

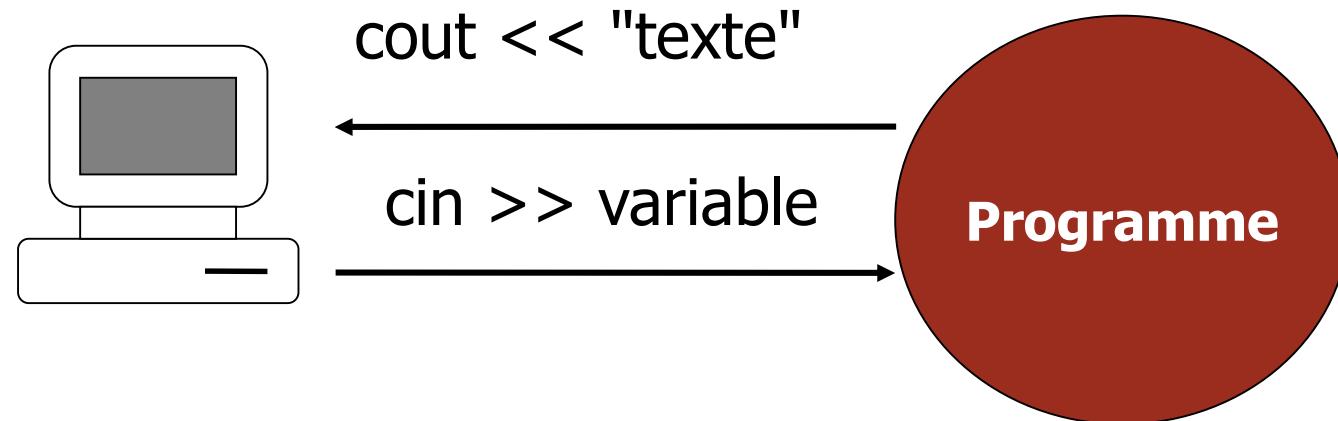
LES ENTRÉES / SORTIES EN C / C++

Communication programme / utilisateur

Traduction de "afficher" et "lire" de l'algorithme

- **cout : permet d'afficher un message à l'écran**
 - Exemple : on veut afficher un message de bienvenue à l'utilisateur :
`cout << "bienvenue" ;`
- **cin : permet de récupérer une valeur fournie par l'utilisateur**
 - Exemple : on veut demander à l'utilisateur une valeur en vue de calculer sa factorielle
`cin >> nomvariable ;`

LES ENTRÉES / SORTIES EN C



Attention :

- << pour cout (du programme vers l'écran)
- >> pour cin (du clavier vers le programme)

LES ENTRÉES / SORTIES EN C

Dans un "cout" on peut mettre

- Des chaînes de caractères : cout <<"bienvenue" ;
- Le contenu de variables : cout << a ;
- Une constante qui permet de passer à la ligne : cout << endl ;

On peut mélanger ces trois types d'affichage dans un même cout ; il suffit de répéter les "<<" :

- Ex : cout <<"bienvenue " << a << endl ;
- si la variable a contient "Pierre" alors on affichera sur la même ligne "bienvenue Pierre" puis on passera à la ligne suivante avec endl.

On peut saisir plusieurs valeurs à la suite dans un "cin" : cin>>a>>b>>c permettra de saisir les trois variables a, b et c.

cin >> "bienvenue" ; est une erreur, pourquoi ?

PLAN

Historique du C

Un programme C

Les entrées / sorties en C

Types des données algorithmique / C

Éléments syntaxiques du langage C

- Structures de contrôles

Traduction d'algorithmes simples en langage C

CORRESPONDANCE DES TYPES

Algorithmique	Langage C / C++
Entier	int, short
Réel	float, double (différence sur la longueur du codage et donc de la précision)
Booleén	bool
Caractère	char

DÉCLARATION DES VARIABLES

algorithmique

- v : entier
- x : réel
- lx : réel
- c : caractère

langage C

- int v ;
- float x ; (4 octets)
- double lx ; (8 octets)
- char c ;

Octet : vecteur de huit bits (chiffre binaire)

pouvant représenter 2^8 (256) valeurs différentes

STOCKAGE DES VARIABLES

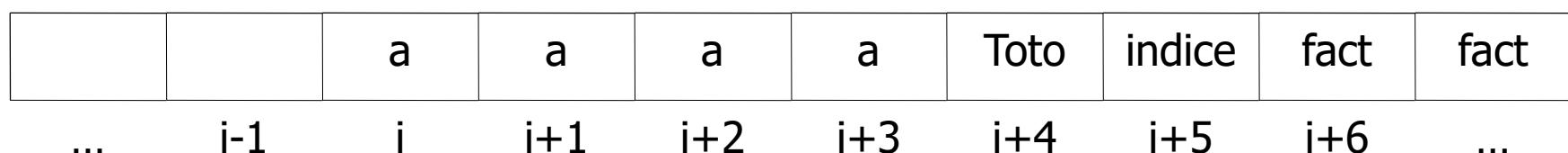
Les variables sont stockées dans la mémoire vive de l'ordinateur

Place occupée fonction du type de la variable

En mémoire, les variables sont repérées par leur emplacement ou adresse

- Soient les déclarations suivantes :
 - int a;
 - char toto, indice;
 - float fact;
- Les variables seront stockées en mémoire de la manière suivante :

Type	Taille (octets)
Char	1
Short, ...	2
Int, long, float	4
Double	8
Long double	10



LES CONSTANTES EN C

Une constante = nom désignant une valeur non modifiable lors de l'exécution d'un programme.

Définie grâce à la directive du préprocesseur **#define**, qui permet de remplacer toutes les occurrences du mot qui le suit par la valeur immédiatement derrière elle.

- `#define PI 3.1415927` remplacera tous les identifiants « PI » (sans les guillemets) par la valeur 3.1415927

Toutefois, avec cette méthode les constantes ne sont pas typées
→ utiliser le mot clé **const**, qui permet de déclarer des constantes typées :

- `const int DIX = 10;`

PLAN

Historique du C

Un programme C

Les entrées / sorties en C

Types des données algorithmique / C

Éléments syntaxiques du langage C

- Structures de contrôles

Traduction d'algorithmes simples en langage C

IDENTIFICATEURS

Nom

- de variable,
- de constantes.

Chaîne de caractères

- commençant par une lettre (majuscule ou minuscule)
- constituée de lettres [a-zA-Z], de chiffres [0-9], de _,
- sans accent, ni espace, ni –

Attention : MAJUSCULES et minuscules différenciées

Convention : nom de constante en majuscule

Un identificateur doit être **évocateur** de ce qu'il représente

- discriminant : identificateur correct pour variable de calcul du discriminant;
- nom_etudiants : identificateur correct...
- Fgmqsdgfk, rapidos, tempo, variable : à éviter !!!

INSTRUCTION SIMPLE

Expression ;

; : à la fin de chaque instruction simple

Exécutées séquentiellement de haut en bas
(sens de la lecture)

L'expression est généralement une affectation

- $x = 3 ;$

BLOC D'INSTRUCTIONS

Bloc : séquence d'instructions entre { ... }

Considéré comme une instruction

À utiliser systématiquement dans les instructions complexes

Instructions exécutées du début à la fin du bloc

Les blocs peuvent être imbriqués

EXEMPLE : PRODUIT DE 2 ENTIERS

Traduction de l'exemple du CM1

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

int main (void)
{
    int a, b, c;
    cout<<"Donnez 2 entiers"<<endl;
    cin>>a>>b;
    c = a*b;
    cout<<"Le produit de "<<a<<" par "<<b<<" est : "<<c<<endl;
    return 0 ;
}
```

STRUCTURES DE CONTRÔLE

Constructions du langage algorithmique

Alternative :

- Si-alors-sinon

Itérations

- Tant que ... Faire ...FinTantQue
- Faire ... Tant que ...
- Pour ... de ... à ... pas de ... Faire ...FinPour

Sélection

- Selon... autrement ... Fin Selon

TRADUCTION DE L'ALTERNATIVE

Si expressionCondition

Alors

Action(s) 1

Sinon

Action(s) 2

FinSi

if (expressionCondition)

{

Action(s) 1

}

else

{

Action(s) 2

}

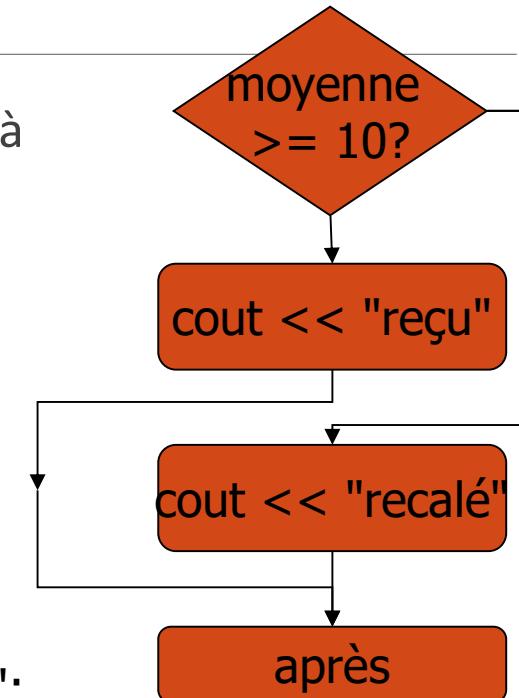
- **Attention** : ne pas mettre de ; après la condition !!
- Comme en algorithmique : partie else (sinon) pas forcément nécessaire

ALTERNATIVE : EXEMPLE

Si la moyenne de deux notes est supérieure ou égal à 10
on affiche reçu sinon on affiche recalé.

```
si (moyenne >=10)
alors
    afficher (reçu)
sinon
    afficher (recalé)
Fin si
```

```
if (moyenne >=10)
{
    cout << "reçu";
} else {
    cout << "recalé";
}
```



TRADUCTION DE TANTQUE...FAIRE

TantQue ExpressionCondition Faire
 Action(s)

FinTantQue

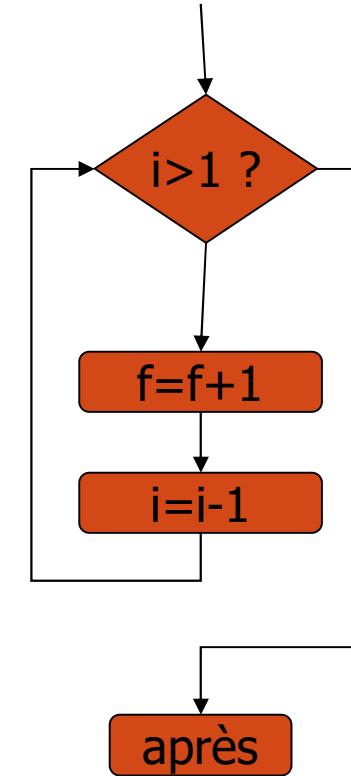
Traduction en C :

```
while (expressionCondition)
{
    Action(s)
}
```

TRADUCTION DE TANTQUE...FAIRE

TantQue $i > 1$ Faire
 $f \leftarrow f + i$
 $i \leftarrow i - 1$
FinTantQue

```
while (i>1)
{
    f=f+i;
    i=i-1;
}
```



TRADUCTION DE FAIRE ... TANTQUE

Faire	do
Action(s)	{
TantQue expressionCondition	Action(s)
	}
	while(expressionCondition);

La séquence d'instructions “Action(s)” est effectuée au minimum une fois puisque l'évaluation de la condition est effectuée au sortir de la boucle.

BOUCLE À NOMBRE D'ITÉRATION CONNU

```
for ( instruction1 ;  
      expressionCondition ;  
      instruction2 )  
{  
    Action(s)  
}
```

```
instruction 1 ; /* intialisation */  
  
/* boucle */  
while ( expressionCondition)  
{  
    Action(s)  
  
    instruction 2 ; /* pas de l'itération */  
}
```

Deux méthodes pour écrire la même chose
Nombre d'itérations connu au départ

EXEMPLE POUR ÉCRIRE LES NOMBRES DE 1 À 10

```
#include <iostream.h>
#include <stdlib.h>
int main()
{
    int i ;
    for (i = 1; i <=10; i=i+1)
    {
        cout << i << " " ;
    }
    return 0 ;
}
```

```
#include <iostream.h>
#include <stdlib.h>
int main()
{
    int i ;
    i = 1;
    while ( i <=10)
    {
        cout << i << " " ;
        i = i + 1 ;
    }
    return 0 ;
}
```

SELON CHOIX

```
switch ( expression )  
{  
    case e1 :  
        Action(s) 1  
        break ;  
    case e2 :  
        Action(s) 2  
        break ;  
    ....  
    default :  
        Action(s) par défaut  
}
```

En algorithmique : selon choix

expression est une expression entière quelconque

break permet de sortir du **switch** ; sinon on continue à exécuter la ligne suivantes

default : au cas où l'expression ne prendrait aucune des valeurs définies

SELON CHOIX : EXEMPLE

selon jour

```
1 : afficher('Lundi')
2 : afficher('Mardi')
3 : afficher('Mercredi')
4 : afficher('Jeudi')
5 : afficher('Vendredi')
6 : afficher('Samedi')
7 : afficher('Dimanche')
```

autrement :

```
afficher('Erreur')
```

fin selon

```
switch (jour)
{
    case 1 : cout<<"lundi";
               break;
    case 2 : cout<<"mardi";
               break;
    case 3 : cout<<"mercredi";
               break;
    case 4 : cout<<"jeudi";
               break;
    case 5 : cout<<"vendredi";
               break;
    case 6 : cout<<"samedi";
               break;
    case 7 : cout<<"dimanche";
               break;
    default : cout<<"erreur";
               break;
}
```

LES COMMENTAIRES EN C

Objectifs

- Expliquer comment fonctionne le programme
- Justifier les choix qui ont été faits
- S'y retrouver quand on reprend un programme

Bloc de commentaires sur plusieurs lignes délimités par /* et */ :
exemple

```
/* blablabla  
ici on calcule ...*/
```

Commentaire en fin de ligne

```
// commentaire
```

À utiliser sans modération !!

EXPRESSIONS ENTIÈRES

Opérations réalisables sur les entiers

Types : int, short

Opérations arithmétiques :

- Opérations mathématiques standards : +, -, *, /
- Modulo = reste de la division entière (%)
 - $(12 \% 5) = 2$
 - $(5 \% 12) = 5$

Résultat entier si opérandes entiers : $1 / 2 == 0$

EXPRESSIONS RÉELLES

Opérations réalisables sur les réels

Types : float, double

- float : stockages
- double : calcul (plus de précision)

Opérations arithmétiques : +, -, *, /

TYPAGE ET CONVERSION IMPLICITE

Calcul :

- 2 opérandes int : résultat int
- 2 opérandes float ou double : résultat double
- opérande entier et opérande réel : résultat double

Affectation :

- entier dans une variable réelle : conversion
 - a=2
 - si a réel alors a=2.0
- réel dans une variable entière : on enlève la partie décimal
 - ent = 2,245
 - si ent entier alors ent = 2 !!!

OPÉRATEURS RELATIONNELS

égal : ==

différent : !=

inférieur ou égal : <=

inférieur strictement : <

supérieur ou égal : >=

supérieur strictement : >

attention : ne pas utiliser == avec les réels

- Ex $((1/3)*3)/3$... égal? 1 ...
- problème de précision dans le codage des réels

OPÉRATEURS LOGIQUES

ET : **&&** (et commercial, perluète ou esperluette)

OU : **||** (pipe, 2 barres verticales) (Alt GR + 6)

NON : **!**

Utiliser les parenthèses pour respecter les priorités des opérateurs

Attention AND / OR et NOT sont des mots clés alternatifs ! Ils PEUVENT se compter comme les opérateurs standards mais certains compilateurs peuvent ne pas les reconnaître ➔ à éviter !

BOOLÉEN

Pas vraiment de booléen en C
(même si le type `bool` existe)

Codage des valeurs booléennes dans les entiers :

- 0 : faux
- autres valeurs, souvent 1 : vrai (`!0 == 1`)

Stockage dans un entier :

- `int b ;`
- `b = ((a < 2) && (i > 10))`
- `b` aura pour valeur soit 0 ou soit 1

OPÉRATIONS EXOTIQUES SUR LES ENTIERS

Opérations qui modifient la valeur stockée

- `++` et `--` (incrémentation et décrémentation automatique)

- souvent utilisé sous la forme : `i++` ;
 - équivalent à `i = i + 1`;

- `a = 0; i = 1; a = i++ ;`

- que valent `a` et `i` après exécution ?
`a == 1, i == 2`

- `a = 0; i = 1; a = ++ i ;`

- que valent `a` et `i` après exécution ?

`a == 2, i == 2`

équivalent

<code>a = i++</code>	<code>a = i</code>
	<code>i = i + 1</code>

équivalent

<code>a = ++i</code>	<code>i = i + 1</code>
	<code>a = i</code>

Opérateurs `+=` `-=` et `/=`

PLAN

Historique du C

Un programme C

Les entrées / sorties en C

Types des données algorithmique / C

Éléments syntaxiques du langage C

- Structures de contrôles

Traduction d'algorithmes simples en langage C

EXEMPLE COMPLET → QUE FAIT CE PROGRAMME ?

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

int main() {
    int n, nombre;

    cout << "Combien de nombres voulez-vous entrer ? ";
    cin >> n;

    cout << "Entrez " << n << " nombres entiers :" << endl;

    for (int i = 1; i <= n; i++) {
        cin >> nombre;

        if (nombre % 2 == 0) {
            cout << nombre << " est pair." << endl;
        } else {
            cout << nombre << " est impair." << endl;
        }
    }

    return 0;
}
```

CONCLUSION

Petit tour d'horizon des éléments syntaxiques de base du langage C

- Types et variables
- Structures de contrôle
- Conditions / expressions

À enrichir durant les prochaines séances

De quoi débuter les travaux pratiques

Présentation rapide de l'outil utilisé