



COLLEGE
DE PARIS
SUPÉRIEUR

DAKARTECH

PROGRAMMATION C

Dr Pape Abdoulaye BARRO

Enseignant – Chercheur

Spécialiste en Télémétrie & Systèmes Intelligents

PLAN

- + Généralités, Types et Operateurs
- + Variables, Lecture/écriture, conditions et expression
- + Structures conditionnelles et itératives
- + Tableaux
- + Sous programmes

VARIABLES, LECTURE/ÉCRITURE, CONDITIONS ET EXPRESSION VARIABLE

- ✖ Une variable est un espace mémoire nommé, de taille fixée, prenant au cours du déroulement d'un programme, un nombre indéfini de valeurs différentes.
 - + Un programme tourne généralement autour des variables;
 - + Le changement de valeur se fait par l'opération d'affectation.
 - + La variable diffère de la notion de **constante** qui, comme son nom l'indique, ne prend qu'une unique valeur au cours de l'exécution du programme.

VARIABLES, LECTURE/ÉCRITURE, CONDITIONS ET EXPRESSION VARIABLE

- ✘ **Toutes les variables doivent être déclarées suivant un type**
 - + `int a, b;`
 - + `float x;`
 - + `char caractere;`
- ✘ **Interdiction d'utiliser les mots-clés (int, break, return, for, etc.)**
- ✘ **Ne doivent pas contenir:**
 - + D'espaces
 - + De caractères accentués,
- ✘ **Commencent par:**
 - + Une lettre
 - + Un « \$ »
 - + Un « _ » (underscore)
- ✘ **Ne commencent pas par:**
 - + Un chiffre
 - + Un signe de ponctuation autre que « \$ » ou « _ »
- ✘ **Contraintes**
 - + Sensibilité à la casse

VARIABLES, LECTURE/ÉCRITURE, CONDITIONS ET EXPRESSION

PORTÉE D'UNE VARIABLE

- ✖ On appelle visibilité ou portée d'une variables les règles qui régissent l'utilisation des variables. Les mêmes règles régissent les types.
- ✖ **Règle 1 : variables globales**
 - + Les variables déclarées avant la 1ere fonction peuvent être utilisées partout. Ces variables sont dites globales.
 - + **Exemple:**

```
#include <stdio.h> //Bibliothèque
int i; //variable globale

void main(){ //début programme
    i=0;
    printf("%d\n",i) //-> 1
} // fin programme
```

VARIABLES, LECTURE/ÉCRITURE, CONDITIONS ET EXPRESSION

PORTÉE D'UNE VARIABLE

✖ Règle 2 : variables locales

+ Les variables déclarées dans une fonction ne peuvent être utilisées que dans cette dernière. Ces variables sont dites locales.

+ Exemple:

```
...  
void process() {  
    int i;  
    i = i+1;  
}  
void main(){  
    i=0; // -> ERREUR : i n'existe dans le main  
    ...  
}
```

VARIABLES, LECTURE/ÉCRITURE, CONDITIONS ET EXPRESSION

PORTÉE D'UNE VARIABLE

- ✖ **Règle 3** : arguments = variables locales
 - + Les arguments d'une fonction sont des variables locales de la fonction.
- ✖ **Règle 4** : Au sein d'une fonction, toutes les variables doivent avoir des noms distincts
- ✖ **Règle 5** : Des variables déclarées dans des fonctions différentes peuvent porter le même nom sans ambiguïté.
- ✖ **Règle 6** : Si une variable globale et une variable locale ont le même nom, on accède à la variable locale dans la fonction où elle est déclarée
- ✖ **Conseil** : Evitez autant que possible l'usage des variables globales

VARIABLES, LECTURE/ÉCRITURE, CONDITIONS ET EXPRESSION

CONSTANTE

- ✖ Une **constante** est une variable particulière dont la valeur ne change pas tout au long de l'exécution du programme
- ✖ Deux types de constantes
 - + Globale
 - ✖ La syntaxe pour définir une constante globale est:
 - ✖ `#define identifiant valeur`
 - ✖ **Exemple** : `#define PI 3,1415`
 - + Locale
 - ✖ La syntaxe pour définir une constante locale est:
 - ✖ `const type identifiant = valeur;`
 - ✖ **Exemple** : `const double PI = 3,1415;`

VARIABLES, LECTURE/ÉCRITURE, CONDITIONS ET EXPRESSION

CONVERSION DE TYPE

✖ Conversion explicite : (type) expression

- + int a; float x; char c;
- + a=2;
- + x=(float) a;
- + x=2.3;
- + a= (int) (x+1);
- + a = 98;
- + c = (char) a; -> c='b'

✖ Conversion implicite

- + int a; float x; char c;
- + a=2;
- + x= a;
- + x=2.3;
- + a= x+1;
- + a = 98;
- + c = a; -> c='b'

VARIABLES, LECTURE/ÉCRITURE, CONDITIONS ET EXPRESSION

LECTURE/ÉCRITURE

- ✗ L'instruction de **lecture** permet de fournir des données à notre programme par l'intermédiaire d'un clavier par exemple.
- ✗ L'instruction d'**écriture** permet à un programme de communiquer des informations ou résultats par l'intermédiaire écran par exemple.
- ✗ Les principales fonctions de lecture et d'écriture en langage C sont **scanf** et **printf** qui sont dans la bibliothèque standard **<stdio>**.
- ✗ La bibliothèque standard **<stdio>** contient un ensemble de fonctions permettant la communication de l'ordinateur avec le monde extérieur.

VARIABLES, LECTURE/ÉCRITURE, CONDITIONS ET EXPRESSION

LECTURE

- ✖ La fonction **scanf** permet de lire à partir du clavier des données.
- ✖ La syntaxe est la suivante:
scanf (“<format> “, <Adrv1>, <Adrv2>, ...);
 - + <format>: format de lecture des données
 - + Autant de format que de données à lire
 - + <Adrv>: &NomVariable
- ✖ La chaîne de format détermine comment les données reçues doivent être interprétées.
- ✖ Les données reçues correctement sont mémorisées successivement aux adresses indiquées par <AdrV1>,... .
- ✖ L'adresse d'une variable est indiquée par le nom de la variable précédé du signe &.

VARIABLES, LECTURE/ÉCRITURE, CONDITIONS ET EXPRESSION

ÉCRITURE

- ✖ La fonction **printf** permet de transférer du texte, des valeurs de variables ou des résultats d'expressions vers l'écran. Elle exige l'utilisation de **formats** de sortie.
- ✖ La syntaxe est la suivante:

```
printf (“<format> “, <expr1>, <expr2>, ...);
```

 - + **<format>**: texte, séquence d'échappement, spécificateur de format
 - + Autant de spécificateurs de formats que d'expressions
 - + Spécificateur de format avec : %caractère_du_type (%d, %f, ...)

VARIABLES, LECTURE/ÉCRITURE, CONDITIONS ET EXPRESSION

SPÉCIFICATEURS DE FORMAT

✕ Voici quelques Spécificateurs de format :

- + %d : entier
- + %c : caractère
- + %f : rationnel en notation décimale
- + %s : chaîne de caractère

Exemples de lecture:

- char alpha;
- int i ;
- float r;
- scanf("%c",&alpha); /* saisie d'un caractère */
- scanf("%d",&i); /* saisie d'un entier en décimal */
- scanf("%x",&i); /* saisie d'un entier en hexadécimal */
- scanf("%f",&r); /* saisie d'un réel */

```
int jour, mois, annee;
scanf("%d %d %d", &jour, &mois, &annee);
```

Exemples d'écriture:

- printf("Bonjour\n");
 - x=100 ; y=x ;
 - printf("La valeur de y est %d\n", y);
 - printf("La somme = %d\n", x+y);
 - moyenne=12.3333 ;
 - printf("La moyenne est %.2f\n", moyenne);
 - c='A' ;
 - printf("Le caractère %c a pour valeur %d", c,c);
- /* va afficher sur l'écran: Le caractère A a le code 65 !
La valeur de c est donc affichée sous deux formats différents.
*/

VARIABLES, LECTURE/ÉCRITURE, CONDITIONS ET EXPRESSION

CONDITION

- ✖ Une condition est une comparaison. Elle est composée de trois éléments :
 - + Une valeur
 - + Un opérateur de comparaison
 - + Une autre valeur
- ✖ Les valeurs peuvent être à priori de n'importe quel type. Mais si l'on veut que la comparaison ait un sens, il faut que les deux valeurs soient du même type et appartiennent à un ensemble ordonné.
- ✖ **Exemple:**
 - + a est plus grand que b

VARIABLES, LECTURE/ÉCRITURE, CONDITIONS ET EXPRESSION COMMENTAIRES

Deux type de commentaires

- × **Mono ligne**

```
//my comment
```

- × **Multi ligne**

```
/*
```

```
my first commented line
```

```
my second commented line
```

```
...
```

```
my nth commented line
```

```
*/
```

VARIABLES, LECTURE/ÉCRITURE, CONDITIONS ET EXPRESSION

INSTRUCTION

- ✖ Une instructions est une opération élémentaire (+,-, *,...) sur des données unitaires.
- ✖ Chaque instruction se termine par un point virgule (;).
- ✖ Le début d'un bloc est signalé par un { et sa fin par }.
- ✖ Un bloc devient l'équivalent syntaxique d'une instruction.

VARIABLES, LECTURE/ÉCRITURE, CONDITIONS ET EXPRESSION

LES DIRECTIVES AU PRÉPROCESSEUR

- ✗ Le **préprocesseur** est un programme exécuté lors de la première phase de la compilation.
- ✗ Il effectue des modifications textuelles sur le fichier source à partir de directives.
- ✗ Les différentes directives au préprocesseur, introduites par le caractère **#**, ont pour but :
 - + l'incorporation de fichiers source (**#include**),
 - + la définition de constantes symboliques et de macros (**#define**),
 - + la compilation conditionnelle (**#if**, **#ifdef**,...).

VARIABLES, LECTURE/ÉCRITURE, CONDITIONS ET EXPRESSION

STRUCTURE D'UN PROGRAMME

- ✖ La ligne **main()** permet de préciser que ce qui sera décrit à sa suite est en faite le **programme principal** (main).
- ✖ Le programme principal vient à la suite des en-tête et est délimité par les accolades « { » et « } ».
- ✖ Les instructions situées entre ces accolades forment un « bloc ».
 - + Notons qu'un bloc peut lui-même contenir d'autres blocs.

✖ Exemple de programme:

```
#include<stdio.h>
int main()
{
    printf("Ceci est votre premier programme\n");
    return 0;
}
```

RAPPEL ALGORITHMIQUE

CAS PRATIQUES N°1

✖ Application 1 :

Ecrivez un programme qui à partir de la valeur saisie du côté d'un carré donné, calcule son périmètre et sa surface et affiche les résultats à l'écran.

✖ Application 2:

Ecrivez un programme qui à partir de la valeur de l'arrête d'un cube saisie au clavier, calcule sa surface de base et son volume et affiche les résultats à l'écran.

✖ Application 3:

Ecrivez un programme qui à partir de la valeur du rayon d'un cercle saisie au clavier, calcule son diamètre, sa surface et sa circonférence.

RAPPEL ALGORITHMIQUE

CAS PRATIQUES N°1

✖ Application 4 :

Ecrivez un programme qui à partir du prix hors taxe PHT d'un produit et du taux de TVA calcule et affiche le prix toute taxe comprise PTTC.

✖ Application 5:

Ecrivez un programme permettant de déclarer trois variables A, B, C de type réel, d'initialiser leurs valeurs et ensuite d'effectuer la permutation circulaire des trois variables.

✖ Application 6:

Ecrire un programme permettant de saisir l'abscisse d'un point A et de calculer son ordonné $f(x) = 2X^3 - 3X^2 + 4$. Evaluer le résultat en expliquant les ordres de priorité pour $x = -2$.

Affaires à suivre



Feedback sur:
pape.abdoulaye.barro@gmail.com