Langage C

Licence 2 D2A et SRT

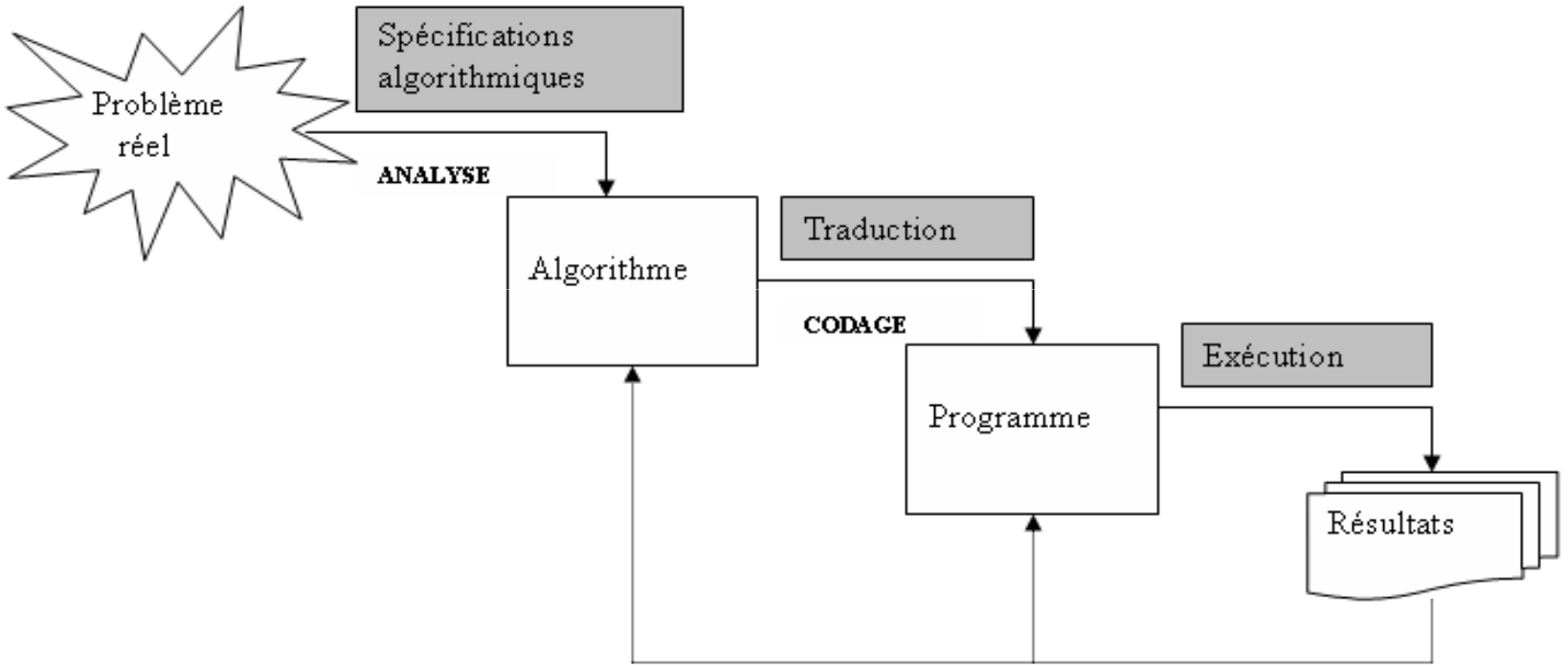
Dr Youssou KASSE
Université Alioune Diop de Bambey
Youssou.kasse@uadb.edu.sn

Concepts de base du Langage C

PLAN

- Historique
- Compilation
- Composants du langage C
- Structure d'un programme C
- Les types de base du langage C
- Déclaration des variables et des constantes
- Les opérateurs du langage C
- Les instructions élémentaires

Rappel: Étapes de la construction d'un programme



Les différentes étapes du processus de programmation

Rappel: Comment bâtir un programme

- En suivant l'une ou l'autre des multiples méthodologies de développement…
- On y retrouve généralement un ensemble de points communs:
  - ***Analyse et définition des besoins*** : ce que l'on s'attend que le programme devra réaliser comme tâche. L'analyse et la spécification du problème se fera dans un ***langage naturel***.
  - ***Spécification du programme*** : définir spécifiquement chaque fonctionnalité du programme.

Rappel:
Comment bâtir un programme

- ***Conception***: décomposition du programme en sous problème (définition des modules du programme) ainsi que le **développement des algorithmes** nécessaires au fonctionnement du programme. La conception des algorithmes s'appuiera sur l'analyse et la spécification du problème.

- ***Programmation***: **traduction des algorithmes** dans le langage de programmation choisi pour la réalisation du programme. **On parle d'implantation**.

Comment bâtir un programme

- ***Tests et validation***: phase durant laquelle le bon fonctionnement, ainsi que la conformité du programme aux spécifications sont vérifiés.
- ***Maintenance***: activité qui assure l'entretien et le bon fonctionnement du programme tant et aussi longtemps qu'il sera utilisé.
- ***Documentation du programme***: chaque étape énoncée doit être documentée pour assurer la pérennité du programme.

Historique du langage C

- Le C a été conçu en 1972 par Dennis Richie et Ken Thompson, chercheurs aux Bell Labs, afin de développer un système d'exploitation UNIX sur un DEC PDP-11.
- En 1978, Brian Kernighan et Dennis Richie publient la définition classique du C dans le livre « *The C Programming language* »
- En1983, l'ANSI (American National Standards Institute) décida de normaliser le langage; ce travail s'acheva en 1989 par la définition de la norme ANSI C

Compilation

- Le C est un langage ***compilé*** *(par opposition aux langages* ***interprétés****).*
- *Cela signifie qu'un* programme C est décrit par un fichier texte, appelé ***fichier source****.*
- *Ce fichier n'étant évidemment* pas exécutable par le microprocesseur, il faut le traduire en langage machine.
- Cette opération est effectuée par un programme appelé ***compilateur****.*

Les langages compilés

- Dans le cas d'un langage compilé (exemples: C, C++, Pascal...) , le programme réalisé, appelé programme source, est traduit complètement par ce qu'on appelle un compilateur avant de pouvoir être exécuté. La compilation génère un programme dit exécutable.

- Ce programme généré est autonome, c'est-à-dire qu'il n'a pas besoin d'un autre programme pour s'exécuter. Mais à chaque modification du fichier source (le programme source) il faudra le recompiler pour que les modifications prennent effet.

Les langages interprétés

- Un programme écrit dans un langage interprété (exemples: Perl, Lisp, Prolog...) a besoin, pour chaque exécution, d'un programme annexe appelé interpréteur qui va lire le code source pour traduire et faire exécuter une à une, chacune des instructions.
- Dans ce cas, il n’y a pas de génération de programme exécutable.

Quelques avantages

- Le langage C est un langage :
  - qui permet (presque) de tout faire,
  - qui permet de créer des exécutables très rapides (compilé),
  - qui possède peu d'instructions,
  - **typé**: tout objet doit avoir un type (caractère, entier, réel...)
  - **déclaratif**: tout objet doit être déclaré avant son utilisation.

Les composants élémentaires du C

- Un programme en langage C est constitué des six groupes de composants élémentaires suivants :
  - les identificateurs,
  - les mots-clefs,
  - les constantes,
  - les chaînes de caractères,
  - les opérateurs,
  - les signes de ponctuation
- Nous pouvons ajouter à ces six les commentaires (/* xxx */ qui sont enlevés par le compilateur

Les identificateurs

- Le rôle d'un identificateur est de donner un nom à une entité du programme.
- Plus précisément, un identificateur peut désigner:
  - un nom de variable ou de fonction,
  - un type défini par **typedef**, **struct**, **union** ou **enum**,

Les identificateurs

- Un identificateur est une suite de caractères parmi:
  - les lettres (minuscules ou majuscules, mais non accentuées),
  - les chiffres,
  - le “blanc souligné” (_).
- Le premier caractère d’un identificateur ne peut être un chiffre.

Les mots-clefs

- Un certain nombre de mots, appelés *mots-clefs*, *sont réservés pour le langage lui-même et* ne peuvent pas être utilisés comme identificateurs.
- L'ANSI C compte 32 mots clefs :
- *Auto, const, double, float, int, short, struct, unsigned, break, continue, else, for, long, signed, switch, void, case, default, enum, goto, register, sizeof, typedef, volatile, char, do, extern, if, return, static, union* et *while.*

Expression en C

- Une ***expression*** *est une suite de composants élémentaires syntaxiquement correcte, par* exemple
  - x = 0
  - ou bien
  - (i >= 0) && (i < 10) && (p[i] != 0)

Instruction en C

- Une ***instruction*** *est une expression suivie d'un point-virgule. Le point-virgule signifie en* quelque sorte « évaluer cette expression ».

- Plusieurs instructions peuvent être rassemblées par des accolades *{ et } pour former une instruction composée ou bloc qui est syntaxiquement é*quivalent à une instruction.
- Exemple: if (cond) *{ ins1; ins 2; ins 3; }*

Déclaration

- Une instruction composée d'un spécificateur de type et d'une liste d'identificateurs séparés par une virgule est une ***déclaration***.
- *Exemple:*
  - int a;
  - int b = 1, c;
  - char message[80];
- En C, toute variable doit faire l'objet d'une déclaration avant d'être utilisée.

Structure d'un programme C

- Un programme en C se compose d'un ***programme principal*** et éventuellement de ***sous programmes***.
- Il est écrit dans un fichier portant l'extension **.c**
- Ce fichier est appelé fichier source du programme et contient généralement les parties suivantes:

Structure d'un programme C

[directives au préprocesseur]
[déclarations de variables externes]
[fonctions secondaires]

main()
{
déclarations de variables internes
instructions
}

Structure d'un programme C

- Les fonctions secondaires peuvent être placées indifféremment avant ou après la fonction principale.
- Une fonction secondaire peut se décrire de la manière suivante :

```
type ma_fonction ( arguments )
{
déclarations de variables internes
instructions
}
```

Structure d'un programme C

- Les directives de compilation: la ligne commence par le symbole # suivi de la directive. Par exemple pour assigner 3.1415926535 au nombre PI, on écrira :
  - #define PI 3.1415926535
  - Mais aussi, pour utiliser une fonction qui a été prédéfinie dans la bibliothèque stdio.h, on met #include <stdio.h>
- définition des structures et des types (s'il y en a)
- déclaration des variables globales (s'il y en a)
- Les fonctions : se sont véritablement les moteurs du programme.
- programme principal (fonction main)

Un premier programme

```
1   /*
2 Mon premier programme C.
3   */
4   #include<stdio.h>
5   int main(void)
 6   {
7      printf("Bonjour tout le monde\n");
8      return 0;
9   }
```

Un premier programme

- [1, 2, 3] Commentaire sur plusieurs lignes.
- [4] **#include** pour pouvoir utiliser la fonction **printf** qui est une fonction de la librairie **stdio.h** (**ST**andar**D** **I**nput/**O**utput).
- [5] Fonction principale, point d'entrée du programme.
- [6] Accolade ouvrante, débute le code de la fonction.
- [7] La fonction printf permet d'afficher un message à l'écran. Les guillemets délimitent la chaîne de caractère. Le caractère **\n** signifie un saut de ligne suivi d'un retour chariot. Le point-virgule à la fin de chaque phrase est indispensable.
- [8] Le programme renvoie la valeur **0** au **shell** appelant à la fin de son exécution.
- [9] Accolade fermante, fin de bloc de la fonction principale

Les directives de compilation

- Elles indiquent au compilateur de procéder à des opérations préalables au début de la compilation.
- Ces directives se situent en tout début du programme source.

La directive #include

- Cette directive permet l'inclusion de librairies dont les éléments seront utilisés dans le programme source.
- Le compilateur C fournit un ensemble de librairies mais le programmeur peut aussi créer ses propres librairies.

- **Syntaxe de cette directive :**
- #include <fichier> ou #include "fichier"
- **Exemples :**
- #include <stdio.h>
- #include "C:\MesProgrammes\definitions.h"

La directive #define

- Cette directive permet de remplacer dans le programme toutes les occurrences d'une suite de caractères par un nom de substitution.
- **Syntaxe de cette directive :**
  - **#define nom_de_substitution suite_de_caractères**
  - *nom_de_substitution* sera utilisé tout au long du programme source pour représenter la suite de caractères *suite_de_caractères*.
- **Exemple:**
- #define VRAI 1
- Le mot VRAI sera utilisé pour représenter le chiffre 1 dans le programme source. Mais rien ne vous empêche d'utiliser le chiffre 1.

Les types prédéfinis

- Le C est un **langage typé**.
- Cela signifie en particulier que toute variable, constante ou fonction est d'un type précis.
- Le type d'un objet définit la façon dont il est représenté en mémoire.
- La taille mémoire correspondant aux différents types dépend des compilateurs; toutefois, la norme ANSI spécifie un certain nombre de contraintes.

Le type caractère

- Le mot-clef **char** désigne un objet de type caractère.
- Un **char** peut contenir n'importe quel élément du jeu de caractères de la machine utilisée.
- La plupart du temps, un objet de **type char** est codé sur un **octet**; c'est l'objet le plus élémentaire en C.
- La plupart des machines utilisent le jeu de caractères ISO-8859 (donc le caractère sur 8 bits)

Le type caractère

- Une des particularités du type char en C est qu'il peut être **assimilé à un entier** : tout objet de type char peut être utilisé dans une expression qui utilise des objets de type entier.
- Par exemple, si c est de type char, l'expression **c + 1 est valide**. Elle désigne le caractère suivant dans le code ASCII.
- Ainsi, le programme suivant imprime le caractère 'B'. *main() { char c = 'A'; printf("%c", c + 1); }*

Les types entiers

- Le mot-clef désignant le type entier est **int**.
- Le type **int** peut être précédé d'un attribut de précision (short ou long).
- Un objet de type **short int** a au moins la taille d'un char et au plus la taille d'un int. En général, un short int est codé sur 16 bits.

Caractère	Entier court	Entier	Entier long
Char	Short	Int	Long
8 bits	16 bits	32 bits	32 bits

Les types flottants

- Les types **float**, **double** et **long double** servent à représenter des nombres en virgule flottante.
- Ils correspondent aux différentes précisions possibles.

Flottant	Flottant double précision	Flottant quadruple précision
Float	Double	Long double
32 bits	64 bits	64 bits

Les constantes

- Une **constante** est une valeur qui apparaît littéralement dans le code source d'un programme.
- Le type de la constante étant déterminé par la façon dont la constante est écrite.
- Déclaration: ***const type identificateur=valeur;***

constante	1234	02322	0x4D2	12.34
Type	Int	int /* octal */	int /* hexadécimal */	double

Les constantes (caractères)

- Pour désigner un caractère imprimable, il suffit de le mettre entre apostrophes (par ex. ’A’ ou ’$’).
- Les seuls caractères imprimables qu’on ne peut pas représenter de cette façon sont l’antislash et l’apostrophe, qui sont respectivement désignés par \\ et \’.
- Toutefois, les caractères non-imprimables les plus fréquents disposent aussi d’une notation plus simple :

\n	\t	\v	\r
nouvelle ligne	tabulation horizontale	tabulation verticale	Retour chariot sans saut de ligne

Les types de base du Langage C

- En langage C, chaque variable doit être associée à un type, qui permet de spécifier la taille de l'espace occupé en mémoire (nombre d'octets).

type	taille	Valeurs possibles	Format
Int	2 ou 4	de -32768 à 32767 de -2147483648 à 2147483647	%d
Short	2	de -32768 à 32767	%d
Long	4	de -2147483648 à 2147483647	%ld
Float	4		%f
Double	8		%lf
Char	1	de -128 à 127	%c

Les opérateurs: L'affectation

- En C, l'**affectation** est un opérateur à part entière.
- Elle est symbolisée par le signe **=**
- Sa syntaxe est la suivante :

variable = expression

- Le terme de gauche de l'affectation peut être une variable simple, un élément de tableau mais pas une constante.

Les opérateurs: L'affectation

variable = expression

- Cette expression a pour effet d'évaluer ***expression*** et d'affecter la valeur obtenue à ***variable***.
- De plus, cette expression possède une valeur, qui est celle expression.
- Ainsi, l'expression i = 5 vaut 5.
- L'affectation effectue une ***conversion de type implicite*** : *la valeur de l'expression (terme de* droite) est convertie dans le type du terme de gauche.

Les opérateurs: L'affectation

- Par exemple, le programme suivant:

```
main()
{
   int i, j = 2;
   float x = 2.5;
   i = j + x;
   x = x + i;
   printf("\n %f \n",x);
}
```

- imprime pour x la valeur **6.5** (et non **7**), car dans l'instruction ***i = j + x;*** ,l'expression ***j + x*** a été convertie en entier.

Les opérateurs arithmétiques

- Les opérateurs arithmétiques classiques sont l'opérateur **unaire -** (changement de signe) ainsi que les opérateurs binaires
  - + addition
  - - soustraction
  - * multiplication
  - / division
  - % reste de la division (modulo)
- L'opérateur **%** ne s'applique qu'à des opérandes de type entier.

Les opérateurs arithmétiques

- Pour l'opérateur **/**, si les deux opérandes sont de type entier, l'opérateur **/** produira une division entière (quotient de la division).
- Par contre, il délivrera une valeur flottante dès que l'un des opérandes est un flottant.
- Par exemple: float x;
- **x = 3 / 2;** affecte à x la valeur 1.
- Par contre **x = 3 / 2.;** affecte à x la valeur 1.5.

Les opérateurs relationnels

- > strictement supérieur
- >= supérieur ou égal
- < strictement inférieur
- <= inférieur ou égal
- == égal
- != différent
- Leur syntaxe est

expression-1 op expression-2

Les opérateurs relationnels

- Les deux expressions sont évaluées puis comparées.
- La valeur rendue est de type int (il n'y a pas de type booléen en C); elle vaut 1 si la condition est vraie, et 0 sinon.
- **Attention** à ne pas confondre l'opérateur de test d'égalité == avec l'opérateur d'affection =.

Les opérateurs logiques booléens

- **&&** et logique
- || ou logique
- **!** négation logique
- Comme pour les opérateurs de comparaison, la valeur retournée par ces opérateurs est un **int** qui vaut 1 si la condition est vraie et 0 sinon.
- Dans une expression de type:

expression-1 op-1 expression-2 op-2 ...expression-n

- L'évaluation se fait de gauche à droite et s'arrête dès que le résultat final est déterminé.

Les opérateurs d'affectation composée

- Les opérateurs d'affectation composée sont

+=	**-=**	***=**	**/=**	**%=**

- Pour tout opérateur op, l'expression:

expression-1 op= expression-2

- est équivalente à:

expression-1 = expression-1 op expression-2

Les opérateurs d'incrémentation et de décrémentation

- Les opérateurs d'**incrémentation ++** et de **décrémentation --** s'utilisent aussi bien en suffixe (i++) qu'en préfixe (++i).
- Dans ces deux cas la variable i sera incrémentée,
- Toutefois dans la notation suffixe la valeur retournée sera l'ancienne valeur de i alors que dans la notation préfixe se sera la nouvelle.

Les opérateurs d'incrémentation et de décrémentation

Pré-incrémentation	**i=++j**	**j=j+1 ; i=j**
Post-incrémentation	i=j++	i=j ; j=j+1
Pré-décrémentation	**i=--j**	**j=j-1 ; i=j**
Post-décrémentation	i=j--	i=j ; j=j-1

- int i,j,k;
- i=1; j=2 ;
- k= i++ + j ; /* i=2, j=2, k=3 */
- k= i + ++j ; /* i=2, j=3, k=5 */
- k= i++ + ++j ; /* i=3, j=4, k=6 */

L'opérateur virgule

- Une expression peut être constituée d'une suite d'expressions séparées par des virgules :

expression-1, expression-2, ... , expression-n

- Cette expression est alors évaluée de gauche à droite. Sa valeur sera la valeur de l'expression de droite.
  - Ce programme
  - **Imprime 5**

```
main()
{
int a, b;
b = ((a = 3), (a + 2));
printf("\n b = %d \n",b);
}
```

- La virgule séparant les arguments d'une fonction ou les déclarations de variables n'est pas l'opérateur virgule.

L'opérateur conditionnel ternaire

- L'opérateur conditionnel **?** est un opérateur ternaire.
- Sa syntaxe est la suivante :

```
condition ? expression-1 : expression-2
```

- Cette expression est égale à ***expression-1*** *si condition est* ***satisfaite***, *et à* ***expression-2*** **sinon**.
- Par exemple:

```
m = ((a > b) ? a : b);
```

- Affecte à m le maximum entre a et b

L'opérateur de conversion de type

- L'opérateur de conversion de type, appelé ***cast***, *permet de modifier explicitement le type* d'un objet.
- On écrit: ***(type) objet***
- Par exemple:

```
main()
{
int i = 3, j = 2;
printf("%f \n",(float)i/j);
}
```

- retourne la valeur **1.5**.

L’opérateur adresse

- L’opérateur d’adresse **&** appliqué à une variable retourne l’adresse-mémoire de cette variable.
- La syntaxe est:

```
&objet
```

Les instructions élémentaires (L'affichage)

- La fonction **printf** permet d'afficher des informations à l'écran
  - **Syntaxe : printf** (" chaîne de caractères " , variable1, variable2, …);
- Cette fonction, contenue dans la bibliothèque standard **stdio.h**, attend comme premier paramètre la chaîne de caractère à afficher avec éventuellement la description des formats d'affichage des variables à afficher
- **Exemple**
  - printf ("la valeur de x est %d et celle de y e st %d", x, y);

Les instructions élémentaires (La lecture)

- L'instruction **scanf** permet au programme de lire des informations saisies au clavier par l'utilisateur.
  - **Syntaxe : scanf** (" chaîne de formatage " , **&**variable1 , **&**variable2, . . .)
- Cette fonction, également contenue dans la bibliothèque standard **stdio.h**, attend comme premier paramètre la chaîne décrivant les formats de lecture des variables à lire au clavier. Les paramètres suivants sont, **dans l'ordre**, **l'adresse** des variables dont on souhaite lire la valeur.
  - **Exemple :** scanf(" %d %d",&X, &Y) ;

Lecture et affichage de caractères

- Les fonctions **getchar** et **putchar** de la bibliothèque **stdio.h** permettent respectivement au programme de lire au clavier et d'afficher à l'écran des caractères.
- Il s'agit de fonctions d'entrées-sorties non formatées.
- **Exemple**: c=getchar(); putchar(c);

QUESTIONS ?