UNIVERSITÉ PRIVÉE DE MARRAKECH

UNIVERSITÉ RECONNUE PAR L'ÉTAT

جامعة معترف بها من طرف الدولة

Module: Structures de données

Chapitre 1: Structures répétitives, conditionnelles, les tableaux, les chaines de caractère

F. OUAKASSE

Introduction

Qu'est ce qu'un algorithme ?

Un ensemble des instructions (programmable sur machine) qui permet de résoudre un problème donné :

- Il lit des données en entrée
- Il effectue un calcul
- Il écrit les résultats de calcul en sortie

Exemple d'algorithmes?

Plan Chapitre 1

1 Structures répétitives (les boucles)

2 Tableaux

3 Chaines de caractères

1 Structures répétitives

- En C, nous disposons de trois structures qui nous permettent la définition de boucles conditionnelles:

1) la structure : **while**
2) la structure : **do - while**
3) la structure : **for**

1 Structures répétitives

- ***La structure while en C***

while (<expression>) <bloc d'instructions>

- * Tant que l'<expression> fournit une valeur différente de zéro, le <bloc d'instructions> est exécuté.
- * Si l'<expression> fournit la valeur zéro, l'exécution continue avec l'instruction qui suit le bloc d'instructions.
- * Le <bloc d'instructions> est exécuté zéro ou plusieurs fois.
- La partie <expression> peut désigner :
  - ✓une variable d'un type numérique,
  - ✓une expression fournissant un résultat numérique.
  - La partie <bloc d'instructions> peut désigner :
    - ✓un (vrai) bloc d'instructions compris entre accolades,
    - ✓une seule instruction terminée par un point-virgule.

1 Structures répétitives

Exemple 1

```
/* Afficher les nombres de 0 à 9 */
int I = 0;
while (I<10)
{
printf("%i \n", I);
I++;
}
```

Exemple 2

```
int I;
/* Afficher les nombres de 0 à 9 */
I = 0;
while (I<10)
printf("%i \n", I++);
 /* Afficher les nombres de 1 à 10 */
 I = 0;
while (I<10)
 printf("%i \n", ++I);
```

Exemple 3

```
/* Afficher les nombres de
10 à 1 */
int I=10;
while (I)
printf("%i \n", I--);
```

1 Structures répétitives

- ***La structure do while en C***

est semblable à la structure **while**, avec la différence suivante :

  - ✓ **while** évalue la condition **avant** d'exécuter le bloc d'instructions,
- **do - while** évalue la condition **après** avoir exécuté le bloc d'instructions. Ainsi le bloc d'instructions est exécuté au moins une fois.
- **do <bloc d'instructions> while (<expression>);**

Le <bloc d'instructions> est exécuté au moins une fois et aussi longtemps que l'<expression> fournit une valeur différente de zéro.

- En pratique, la structure **do - while** n'est pas si fréquente que **while**; mais dans certains cas, elle fournit une solution plus élégante. Une application typique de **do** - **while** est la saisie de données qui doivent remplir une certaine condition.

1 Structures répétitives

- ***Exemple 1***
- float N;
- do { printf("Introduisez un nombre entre 1 et 10 :");
- scanf("%f", &N); }
- while (N<1 || N>10);
- ***Exemple 2***
- int n, div;
- printf("Entrez le nombre à diviser : ");
- scanf("%i", &n);
- do {
- printf("Entrez le diviseur (0) : ");
- scanf("%i", &div); }
- while (!div);
- printf("%i / %i = %f\n", n, div, (float)n/div);

1 Structures répétitives

- **La structure for en C**

for (<expr1> ; <expr2> ; <expr3>)

- **<bloc d'instructions>** est équivalent à : **<expr1>; while (<expr2>) { <bloc d'instructions> <expr3>; } <expr1>** est évaluée une fois avant le passage de la boucle.
  Elle est utilisée pour initialiser les données de la boucle.**<expr2>** est évaluée avant chaque passage de la boucle.
  Elle est utilisée pour décider si la boucle est répétée ou non.
- **<expr3>** est évaluée à la fin de chaque passage de la boucle.
  Elle est utilisée pour réinitialiser les données de la boucle.
- Le plus souvent, **for** est utilisé comme boucle de comptage :
- **for (<init.> ; <cond. répétition> ; <compteur>) <bloc d'instructions>**

1 Structures répétitives

- ***Exemple***
- int I;
- for (I=0 ; I<=20 ; I++)
- printf("Le carré de %d est %d \n", I, I*I);
- En pratique, les parties <expr1> et <expr2> contiennent souvent plusieurs initialisations ou réinitialisations, *séparées par des virgules*.
- ***Exemple***
- int n, t;
- for (t=0, n=1 ; n<101 ; n++)
- t+=n;
- printf("La somme des nombres de 1 à 100 est %d\n", t);

Rappel

Opérateurs arithmétiques

+	addition
-	soustraction
*	multiplication
/	division (entière et rationnelle!)
%	modulo (reste d'une div. entière)

Opérateurs logiques

&&	et logique (and)
||	ou logique (or)
!	négation logique (not)

Opérateurs de comparaison

==	égal à
!=	différent de
<, <=, >, >=	plus petit que, ...

Opérateurs d'affectation

+=	ajouter à
-=	diminuer de
*=	multiplier par
/=	diviser par
%=	modulo

Rappel

Spécificateurs de format pour printf

SYMBOLE	*TYPE*	*IMPRESSION COMME*
`%d ou %i`	`int`	entier relatif
`%u`	`int`	entier naturel (unsigned)
`%o`	`int`	entier exprimé en octal
`%x`	`int`	entier exprimé en hexadécimal
`%c`	`int`	caractère
`%f`	`double`	rationnel en notation décimale
`%e`	`double`	rationnel en notation scientifique
`%s`	`char*`	chaîne de caractères

`I++` ou `++I`	pour l'incrémentation	(augmentation d'une unité)
`I--` ou `--I`	pour la décrémentation	(diminution d'une unité)

Rappel

Spécificateurs de format pour scanf

SYMBOLE	*LECTURE D'UN(E)*	*TYPE*
`%d ou %i`	entier relatif	`int*`
`%u`	entier naturel (unsigned)	`int*`
`%o`	entier exprimé en octal	`int*`
`%b`	entier exprimé en hexadécimal	`int*`
`%c`	caractère	`char*`
`%s`	chaîne de caractères	`char*`
`%f ou %e`	rationnel en notation décimale ou exponentielle (scientifique)	`float*`

Soient les instructions:
int A,B;
scanf("%4d %2d", &A, &B);
Si nous entrons le nombre **1234567**, nous obtiendrons les affectations suivantes: **A=1234**
B=56

Exercice

Calculez la factorielle N! = 123...(N-1)N d'un entier naturel N en respectant que 0!=1.

- a) Utilisez **while**,
- b) Utilisez **for**.

2 Tableaux

Tableau à une dimension

- ***Définitions***

- Un tableau (uni-dimensionnel) A est une variable structurée formée d'un nombre entier N de variables simples du même type, qui sont appelées les ***composantes*** du tableau. Le nombre de composantes N est alors la ***dimension*** du tableau.
- En faisant le rapprochement avec les mathématiques, on dit encore que *"A est un **vecteur** de dimension N »*

- ***Exemple***

- La déclaration
- **int JOURS[12]={31,28,31,30,31,30,31,31,30,31,30,31};** définit un tableau du type **int** de dimension 12. Les 12 composantes sont initialisées par les valeurs respectives 31, 28, 31, ... , 31.On peut accéder à la première composante du tableau par JOURS[0], à la deuxième composante par JOURS[1], . . . , à la dernière composante par JOURS[11].

2 Tableaux

Déclaration de tableaux en langage algorithmique

Déclaration de tableaux en C

<TypeSimple> <NomTableau>[<Dimension>];

Exemples

Les déclarations suivantes en langage algorithmique,

int A[25];

ou bien

long A[25];

ou bien

float B[100];

ou bien

double B[100];

char D[30];

2 Tableaux

Tableau à 2 dimensions

- ***Définitions***
- En C, un tableau à deux dimensions A est à interpréter comme un tableau (uni-dimensionnel) de dimension L dont chaque composante est un tableau (uni-dimensionnel) de dimension C.
- On appelle L le ***nombre de lignes*** du tableau et C le ***nombre de colonnes*** du tableau. L et C sont alors les deux ***dimensions*** du tableau. Un tableau à deux dimensions contient donc ***L*C composantes***.
- On dit qu'un tableau à deux dimensions est ***carré***, si L est égal à C.
- En faisant le rapprochement avec les mathématiques, on peut dire que *"A est un vecteur de L vecteurs de dimension C"*, ou mieux:
- *"A est une **matrice** de dimensions L et C"*.

2 Tableaux

- **Saisie**

```
main()
{
int A[5][10];
int I,J; /* Pour chaque ligne ... */
for (I=0; I<5; I++)
 /* ... considérer chaque
composante */
for (J=0; J<10; J++)
scanf("%d", &A[I][J]);
return 0;
}
```

Rq: Pour obtenir des colonnes bien alignées lors de l'affichage, nous pouvons utiliser la chaîne de format "**%7d**"

- **Affichage**

```
main()
{
int A[5][10];
int I,J; /* Pour chaque ligne ... */
for (I=0; I<5; I++)
{
/* ... considérer chaque composante */
for (J=0; J<10; J++)
printf("%7d", A[I][J]); /* Retour à la ligne */
printf("\n");
}
return 0;
}
```

3 Les chaines de caractères

- ***Déclaration de chaînes de caractères en C***
- char <NomVariable> [<Longueur>];
- ***Exemples***
- char NOM [20];
- char PRENOM [20];
- char PHRASE [300];
- ***Espace à réserver***
- Lors de la déclaration, nous devons indiquer l'espace à réserver en mémoire pour le stockage de la chaîne.
- La représentation interne d'une chaîne de caractères est terminée par le symbole **'\0'** (NUL). Ainsi, pour un texte de **n** caractères, nous devons prévoir **n+1** octets.
- Malheureusement, le compilateur C ne contrôle pas si nous avons réservé un octet pour le symbole de fin de chaîne; l'erreur se fera seulement remarquer lors de l'exécution du programme ...

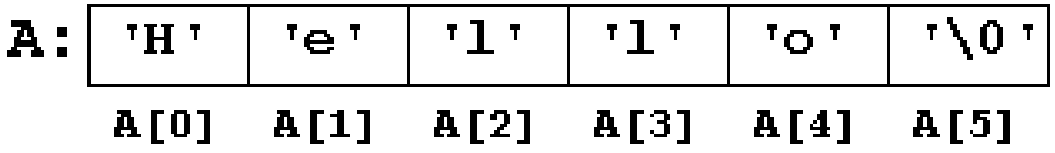
3 Les chaines de caractères

- ***Initiation de chaîne de caractères***
- En général, les tableaux sont initialisés par l'indication de la liste des éléments du tableau entre accolades:
- **char CHAINE[] = {'H','e','l','l','o','\0'};** Pour le cas spécial des tableaux de caractères, nous pouvons utiliser une initialisation plus confortable en indiquant simplement une chaîne de caractère constante:
- **char CHAINE[] = "Hello";** Lors de l'initialisation par **[]**, l'ordinateur réserve automatiquement le nombre d'octets nécessaires pour la chaîne, c.-à-d.: le nombre de caractères **+ 1** (ici: 6 octets). Nous pouvons aussi indiquer explicitement le nombre d'octets à réserver, si celui-ci est supérieur ou égal à la longueur de la chaîne d'initialisation.

3 Les chaines de caractères

- L'accès à un élément d'une chaîne de caractères peut se faire de la même façon que l'accès à un élément d'un tableau. En déclarant une chaîne par:
- **char A[6];** nous avons défini un tableau A avec six éléments, auxquels on peut accéder par: **A[0], A[1], ... , A[5]**
- ***Exemple***

```
char A[6] = "Hello";
```



3 Les chaines de caractères

- Les chaînes de caractères constantes sont indiquées entre guillemets. La chaîne de caractères vide est alors: **"“**
- Dans les chaînes de caractères, on peut utiliser toutes les séquences d'échappement définies comme caractères constants:

"Ce \ntexte \nsera réparti sur 3 lignes."

- Le symbole **"** peut être représenté à l'intérieur d'une chaîne par la séquence d'échappement

\": **"Affichage de \"guillemets\" \n"**

- Le symbole **'** peut être représenté à l'intérieur d'une liste de caractères par la séquence d'échappement **\'** :
  {'L','\'','a','s','t','u','c','e','\0'}
- Plusieurs chaînes de caractères constantes qui sont séparées par des signes d'espacement (espaces, tabulateurs ou interlignes) dans le texte du programme seront réunies en une seule chaîne constante lors de la compilation:**"un " "deux" " trois"**sera évalué à **"un deux trois"**

3 Les chaines de caractères

Exemples de declaration

a) `char a[] = "un\ndeux\ntrois\n";`
b) `char b[12] = "un deux trois";`
Déclaration incorrecte: la chaîne d'initialisation dépasse le bloc de mémoire réservé.
Correction: `char b[14] = "un deux trois";` ou : `char b[] = "un deux trois";`
c) `char c[] = 'abcdefg';`
Déclaration incorrecte: pour initialiser avec une chaîne de caractères, il faut utiliser les guillemets
(ou indiquer une liste de caractères). Correction: `char c[] ="abcdefg";`
d) `char d[10] = 'x';`
Déclaration incorrecte: Il faut utiliser une liste de caractères ou une chaîne pour l'initialisation
Correction: `char d[10] ={'x', '\0'}` ou : `char d[10] ="x";`
e) `char e[5] = "cinq";`
f) `char f[] = "Cette " "phrase" "est coupée";`
g) `char g[2] = {'a', '\0'};`
h) `char h[4] = {'a', 'b', 'c'};`
Déclaration incorrecte: il faut aussi indiquer le symbole de fin de chaîne.
Correction: `char h[4] = {'a', 'b', 'c', '\0'};`

3 Les chaines de caractères

Affichage d'une chaine de caractères

- la bibliothèque <*stdio*> nous offre des fonctions qui effectuent l'entrée et la sortie des données.
- A côté des fonctions **printf** et **scanf** que nous connaissons déjà, on a les deux fonctions **puts** et **gets,** spécialement conçues pour l'écriture et la lecture de chaînes de caractères.
- **printf** avec le spécificateur de format **%s** permet d'intégrer une chaîne de caractères dans une phrase.
- Ds **puts** est idéale pour écrire une chaîne constante ou le contenu d'une variable dans une ligne isolée.

Syntaxe: **puts(<Chaîne>)**

Effet: **puts** écrit la chaîne de caractères désignée par <Chaîne> sur *stdout* et provoque un retour à la ligne. En pratique,

puts(TXT); est équivalent à **printf("%s\n",TXT);**

3 Les chaines de caractères

Lecture d'une chaine de caractères

- **scanf** avec le spécificateur **%s** permet de lire un mot isolé à l'intérieur d'une suite de données du même ou d'un autre type.

Effet: **scanf** avec le spécificateur **%s** lit un ***mot*** du fichier d'entrée standard *stdin* et le mémorise à l'adresse qui est associée à **%s**.

Exemple

```
char LIEU[25];
int JOUR, MOIS, ANNEE; printf("Entrez lieu et date de
naissance : \n");
scanf("%s %d %d %d", LIEU, &JOUR, &MOIS, &ANNEE);
```

- **gets** est idéal pour lire une ou plusieurs lignes de texte (p.ex. des phrases) terminées par un retour à la ligne.

Syntaxe: **gets(<Chaîne>)**

Effet: **gets** lit une ***ligne*** de caractères de *stdin* et la copie à l'adresse indiquée par <Chaîne>. Le retour à la ligne final est remplacé par le symbole de fin de chaîne **'\0'**.

3 Les chaines de caractères

les fonctions de string

La bibliothèque <*string*> fournit une multitude de fonctions pratiques pour le traitement de chaînes de caractères.
Dans le tableau suivant, <n> représente un nombre du type **int**. Les symboles <s> et <t> peuvent être remplacés par :

- une chaîne de caractères constante
- le nom d'une variable déclarée comme tableau de **char**
- un pointeur sur **char** (chapitre suivant)

3 Les chaines de caractères

les fonctions de string

strlen(<s>)	fournit la longueur de la chaîne ***sans*** compter le '\0' final
strcpy(<s>, <t>)	copie <t> vers <s>
strcat(<s>, <t>)	ajoute <t> à la fin de <s>
strcmp(<s>, <t>)	compare <s> et <t> lexicographiquement et fournit un résultat: Négatif si <s> précède <t> Zéro si <s> est égal à <t> Positif si <s> suit <t>
strncpy(<s>, <t>, <n>)	copie au plus <n> caractères de <t> vers <s>
strncat(<s>, <t>, <n>)	ajoute au plus <n> caractères de <t> à la fin de <s>