Les Structures de Contrôle

Structures de contrôle

- Les structures de contrôle définissent la façon avec laquelle les instructions sont effectuées. Elles conditionnent l'exécution d'instructions à la valeur d'une expression

- On distingue :

  - **Les structures alternatives (tests) :** permettent d'effectuer des choix càd de se comporter différemment suivant les circonstances (valeur d'une expression). En C, on dispose des instructions : ***if…else*** et ***switch.***

  - **Les structures répétitives (boucles) :** permettent de répéter plusieurs fois un ensemble donné d'instructions. Cette famille dispose des instructions : **w*hile, d*o…while** et ***for.***

L'instruction if…else

- **Syntaxe :** ***If*** *(expression)*
  bloc-instruction1
  else
  bloc-instruction2

  - *bloc-instruction* peut être une seule instruction terminée par un point-virgule ou une suite d'instructions délimitées par des accolades{ }

  - *expression* est évaluée, si elle est vraie (valeur différente de 0), alors *bloc-instruction1* est exécuté. Si elle est fausse (valeur 0) alors *bloc-instruction2* est exécuté

- La partie *else* est facultative. S'il n'y a pas de traitement à réaliser quand la condition est fausse, on utilisera simplement la forme :

  If *(expression)*
  bloc-instruction1

if…else : exemple

```
float a, b, max;
    if (a > b)
        max = a;
    else
        max = b;
```

Imbrication des instructions if

- On peut imbriquer plusieurs instructions if…else
- Ceci peut conduire à des confusions, par exemple :
  - **if (N>0)**
    if (A>B)
    MAX=A;
    else MAX=B; (interprétation 1 : si N=0 alors MAX prend la valeur B)
  - **if (N>0)**
    if (A>B)
    MAX=A;
    else MAX=B; (interprétation 2 : si N=0 MAX ne change pas)
- En C un *else* est toujours associé au dernier *if* qui ne possède pas une partie *else* (c'est l'interprétation 2 qui est juste)

Imbrication des instructions if

- Conseil : pour éviter toute ambiguïté ou pour forcer une certaine interprétation dans l'imbrication des ***if***, il vaut mieux utiliser les accolades

```
if(a<=0)
    {if(a==0)
        printf("a est nul ");
        else
        printf(" a est strictement négatif ");
        }
else
   printf(" a est strictement positif " );
```

- Pour forcer l'interprétation 1:

```
if (N>0)
        { if (A>B)
            MAX=A;
          }
    else MAX=B;
```

L'instruction d'aiguillage switch :

- Permet de choisir des instructions à exécuter selon la valeur d'une expression qui doit être de type entier

  - la syntaxe est :

    switch *(expression) {*

    case *expression_constante1 : instructions_1;* ***break;***

    case *expression_constante2 : instructions_2;* ***break;***

    …

    case *expression_constante n : instructions_n;* ***break;***

    default *: instructions;*

    }

  - *expression_constantei* doit être une expression constante **entière**
  - Instructions i peut être une instruction simple ou composée
  - *break* et *default sont* optionnels et peuvent ne pas figurer

Fonctionnement de switch

- *expression est évaluée*
- *si sa valeur est égale à une expression_constante i, on se branche à ce cas et on exécute les instructions_i* qui lui correspondent
  - On exécute aussi les instructions des cas suivants jusqu'à la fin du bloc ou jusqu'à une instruction break (qui fait sortir de la structure switch)
- si la valeur de l'expression n'est égale à aucune des expressions constantes
  - Si **default** existe, alors on exécute les instructions qui le suivent
  - Sinon aucune instruction n'est exécutée

Switch : exemple

```
main( )
{ char c;
switch (c) {
        case 'a':
        case 'e':
        case 'i':
        case 'o':
        case 'u':
        case 'y': printf("voyelle\n");break ;
        default : printf("consonne\n");
        }
}
```

Les boucles while et do .. while

```
while (condition)
    {
        instructions
    }
```

```
do
    {
        instructions
    } while (condition);
```

- la condition (dite condition de contrôle de la boucle) est évaluée à chaque itération. Les instructions (corps de la boucle) sont exécutés tant que la condition est vraie, on sort de la boucle dès que la condition devient fausse
- dans la boucle while le test de continuation s'effectue avant d'entamer le corps de boucle qui, de ce fait, peut ne jamais s'exécuter
- par contre, dans la boucle do-while ce test est effectué après le corps de boucle, lequel sera alors exécuté au moins une fois

Boucle while : exemple

Un programme qui détermine le premier nombre entier N tel que la somme de 1 à N dépasse strictement 100

```
main( )
{    int i, som;
      i =0; som= 0;
     while (som <=100)
            {
            som+=i;
            i++;
            }
     printf(" La valeur cherchée est N= %d\n ", i);
}
```

Boucle do .. while : exemple

Contrôle de saisie d'une note saisie au clavier jusqu'à ce que la valeur entrée soit valable

```
main()
   { int N;
        do {
             printf (" Entrez une note comprise entre 0 et 20 \n");
             scanf("%d",&N);
           } while (N < 0 || N > 20);
   }
```

La boucle for

```
for (expr1 ; expr2 ; expr3)
    {
       instructions
    }
```

- L'expression expr1 est évaluée une seule fois au début de l'exécution de la boucle. Elle effectue l'initialisation des données de la boucle

- L'expression expr2 est évaluée et testée avant chaque passage dans la boucle. Elle constitue le test de continuation de la boucle.

- L'expression expr3 est évaluée après chaque passage. Elle est utilisée pour réinitialiser les données de la boucle

Boucle for : remarques

for (expr1 ; expr2 ; expr3) équivaut à :

```
{
   instructions
}
```

```
expr1;
while(expr2)
{ instructions
  expr3;
}
```

- En pratique, expr1 et expr3 contiennent souvent plusieurs initialisations ou réinitialisations, *séparées par des virgules*

Boucle for : exemple

Calcul de x à la puissance n où x est un réel non nul et n un entier positif ou nul

```
main ( )
{     float x, puiss;
      int n, i;
      { printf (" Entrez respectivement les valeurs de x et n \n");
        scanf ("%f %d" , &x, &n);
        for (puiss =1, i=1; i<=n; i++)
              puiss*=x;
        printf (" %f à la puissance %d  est égal à : %f", x,n,puiss);
      }
}
```

L'instruction break

- L'instruction break peut être utilisée dans une boucle (for, while, ou do .. while). Elle permet d'arrêter le déroulement de la boucle et le passage à la première instruction qui la suit

- En cas de boucles imbriquées, break ne met fin qu' à la boucle la plus interne

```
{int i,j;
    for(i=0;i<4;i++)
        for (j=0;j<4;j++)
          { if(j==1) break;
            printf("i=%d,j=%d\n ",i,j);
           }
}
```

résultat:

```
i=0,j=0
i=1,j=0
i=2,j=0
i=3,j=0
```

L'instruction continue

- L'instruction continue peut être utilisée dans une boucle (for, while, ou do .. while). Elle permet l'abandon de l'itération courante et le passage à l'itération suivante

```
{int i;
   for(i=1;i<5;i++)
     {printf("début itération %d\n " ,i);
          if(i<3) continue;
          printf(" fin itération %d\n " ,i);
     }
}
```

résultat:

```
début itération 1
début itération 2
début itération 3
  fin itération 3
début itération 4
  fin itération 4
```

Les Tableaux et les chaines de caractères

Tableaux

- Un **tableau** est une variable structurée composée d'un nombre de variables simples de même type désignées par un seul identificateur
- Ces variables simples sont appelées *éléments ou composantes* du tableau, elles sont stockées en mémoire à des emplacements contigus (l'un après l'autre)

- Le type des éléments du tableau peut être :
  - simple : char, int, float, double, …
  - pointeur ou structure (chapitres suivants)

- On peut définir des tableaux :
  - à une dimension (tableau unidimensionnel ou vecteur)
  - à plusieurs dimensions (tableau multidimensionnel)

Déclaration des tableaux

- La déclaration d'un tableau à une dimension s'effectue en précisant le type de ses éléments et sa dimension (le nombre de ses éléments) :
  - Syntaxe en C : **Type identificateur[dimension];**
  - Exemple : **float notes[30];**
- La déclaration d'un tableau permet de lui réserver un espace mémoire dont la taille (en octets) est égal à : dimension * taille du type
- ainsi pour :
  - **short A[100];** // on réserve 200 octets (100* 2octets)
  - **char mot[10];** // on réserve 10 octets (10* 1octet)

Déclaration des tableaux

- Déclarer un tableau de 5 entiers
- Un tableau de 20 caractères
- Un tableau de 100 nombres réelles en simple précision
- Un tableau de 100 nombres réelles en double précision

```
int t[5];
char  a[20];
float  x[100];
double y[100];
```

Déclaration des tableaux

- En C on peut factoriser par le même type identique lors de la déclaration de plusieurs tableaux de même type. Par exemple, on peut faire les déclarations suivantes:
  - int a[10], b[20];
  - float x[100], y[200];

Que déclare – t-on par les instructions suivantes:

```
int a, b[100];
char  c[15], d[10];
float x[10], z[20];
```

Déclaration des tableaux

- a : un entier simple
- b : un tableau de 100 entiers
- c: un tableau de 15 caractères
- d: un tableau de 10 caractères
- x : un tableau de 10 réels simple précision
- y : un tableau de 20 réels simple précision

Initialisation à la déclaration

- On peut initialiser les éléments d'un tableau lors de la déclaration, en indiquant la liste des valeurs respectives entre accolades. Ex:
  - int A[5] = {1, 2, 3, 4, 5};
  - float B[4] = {-1.5, 3.3, 7e-2, -2.5E3};

- Si la liste ne contient pas assez de valeurs pour toutes les composantes, les composantes restantes sont initialisées par zéro

  - Ex: short T[10] = {1, 2, 3, 4, 5};

- la liste ne doit pas contenir plus de valeurs que la dimension du tableau.
  Ex: short T[3] = {1, 2, 3, 4, 5}; ➔ Erreur

- Il est possible de ne pas indiquer la dimension explicitement lors de l'initialisation. Dans ce cas elle est égale au nombre de valeurs de la liste. Ex: short T[ ] = {1, 2, 3, 4, 5};➔ tableau de 5 éléments

Initialisation à la déclaration

- Indiquer les éléments de chacun des tableaux suivants, ainsi que leur nombre:
  - int tab1[15]={2, 3,5,7,11};
  - int tab2[]={0,2,4,6,8};
  - int tab3[100]={1};
- Découvrir les erreures dans les déclarations suivantes:
  - int []={2, 3,5,7,11};
  - int a[3]={10,20,40,60,80};
  - int s[-4];
  - int t[4,7];

- Indiquer les éléments de chacun des tableaux suivants, ainsi que leur nombre:

    - tab1 contient 5 éléments qui sont 2,3,5,7 et 11
    - tab2 contient 5 éléments qui sont 0,2,4,6 et 8
    - tab3 contient 100 éléments le premier élément de tab3 est 1 et les autres sont tous égaux à 0

- Découvrir les erreurs dans les déclarations suivantes:

    - Le nom du tableau n'est pas indiqué
    - Le nombre de valeurs dans la séquence d'initialisation dépasse le nombre des éléments indiqué (qui est 3)
    - Le nombre des éléments d'un tableau ne doit pas être négatif
    - Le nombre des éléments d'un tableau ne doit pas être un nombre avec virgule

Accès aux composantes d'un tableau

- L'accès à un élément du tableau se fait au moyen de l'indice. Par exemple, **T[i]** donne la valeur de l'élément i du tableau T

- En langage C l'indice du premier élément du tableau est 0. L'indice du dernier élément est égal à la dimension-1

  - Ex: int T[5] = {9, 8, 7, 6, 5}; ➔ T[0]=9, T[1]=8, T[2]=7, T[3]=6, T[4]=5

Remarques:

- on ne peut pas saisir, afficher ou traiter un tableau en entier, ainsi on ne peut pas écrire printf(" %d",T) ou scanf(" %d",&T)

- On traite les tableaux élément par élément de façon répétitive en utilisant des boucles

Tableaux : saisie et affichage

- Saisie des éléments d'un tableau T d'entiers de taille n :

```
for(i=0;i<n;i++)
    { printf ("Entrez l'élément %d \n ",i + 1);
     scanf(" %d" , &T[i]);
    }
```

- Affichage des éléments d'un tableau T de taille n :

```
for(i=0;i<n;i++)
    printf (" %d \t",T[i]);
```

Tableaux : exemple

- Calcul du nombre d'étudiants ayant une note supérieure à 10 :

```
main ( )
   {     float notes[30];  int nbre,i;
         for(i=0;i<30;i++)
              { printf ("Entrez notes[%d] \n ",i);
                scanf(" %f" , &notes[i]);
              }
            nbre=0;
            for (i=0; i<30; i++)
if (notes[i]>10) nbre+=1;
            printf (" le nombre de notes > à 10 est égal à : %d", nbre);
   }
```