Séance 2 : Saisie et instructions de contrôle

Langage C : séance 2

Les Entrées/Sorties :

- Saisie clavier :

  - On utilise la fonction « scanf » de la librairie standard du langage C.

  ➔ Demander à utiliser la librairie standard des E/S :

  #include <stdio.h> // à mettre en début de fichier

  - Paramètres de la fonction « scanf » :

  int i=8;

  scanf("%d", & i);

  & ➔ opérateur d'adresse

  - Premier paramètre : un format à spécifier pour chaque type de variable (idem printf).

  - !!!! Le deuxième paramètre (ici &i) correspond à l'adresse mémoire où se trouve l'entier.
  - !!!! Le deuxième paramètre doit toujours correspondre à l'adresse où l'on doit écrire la donnée.
  - valeur retournée par scanf : le nombre d'affectations correctes.

Langage C : séance 2

Saisie de chaînes de caractères avec « scanf » :

Si ch est une chaîne de caractères : *char ch[10];*

- Saisie limitée de caractères :

  scanf(‘’ %5s‘’ , ch);

  Avec 5, nombre max de caractères à saisir.

- Saisie comprenant un (ou plusieurs) espace(s) :

  scanf(‘’%[^\n]’’, ch);

- Saisie avec espace et 5 caractères max par exemple :

  scanf(‘’%5[^\n]’’, ch); // avec 5 nombre max de car.

Langage C : séance 2

La fonction « sscanf » :

- Permet d'extraire des données à partir d'une chaînes de caractères.
- Syntaxe : sscanf(*dans , comment , où*);

  Recherche l'information dans le premier argument, sous la forme du comment, et le range dans le où.

La fonction « sscanf » retourne le nombre de valeurs extraites correctement.

- exemple :

```
int entier;
float reel;
char tampon[51];
…
printf(''entrez le message \n'');          // on tape :   «12   3.14»
scanf(''%50[^\n]'', tampon);
sscanf(tampon,''%d %f'',&entier,&reel);      //  entier vaut 12, reel vaut 3.14
```

Langage C : séance 2

Saisie d'un caractère : la fonction « getchar » :

Elle appartient à la bibliothèque « stdio.h ».

Usage :

```
char car;
car=getchar();  // équivalent à scanf("%c", &car);
```

Langage C : séance 2

La fonction « sprintf » :

- Permet de convertir en chaîne de caractères des données numériques.
- Syntaxe : sprintf(chaine résultat *, format , variables*);

  Recherche l'information dans le(s) dernier(s) argument(s), sous la forme du format, et le stocke dans la chaîne résultat.

- exemple :

```
int entier=10;
float reel=3.14;
char tampon[50];
…
sprintf(tampon,''%d %f'',entier,reel);
printf(''%s'', tampon);  // affiche la chaîne ''10 3.14''
```

- Privilégier l'usage de **snprintf :**

```
snprintf(tampon,50, ''%d %f'',entier,reel);   // évite le débordement du tampon (50 max ici)
```

Langage C : séance 2

L'instruction « switch » de base :

Syntaxe :

```
switch (expression_selection)
        {
         case valeur1 :  i1;i2;…; in;
         case valeur2 :  i21;i22;…; i2n;
        …
        default   :   idef1;idef2; … ;
        }
```

facultatif

Exemple :

```
int i=1;
switch ( i )
        {
         case 1:  printf("valeur 1 ");
         case 2 :  printf("valeur 2 ");
        default   :   printf("valeur defaut ");
        }
```

Affichage ➔ valeur 1 valeur 2 valeur defaut

Langage C : séance 2

L'instruction « switch » : usage très fréquent.

Syntaxe :

```
switch (expression_selection)
        {
         case valeur1 :  i1;i2;…; in; break;
         case valeur2 :  i21;i22;…; i2n; break
        …
        default   :   idef1;idef2; … ;
        }
```

L'instruction break permet de sortir du bloc switch

Exemple :

```
int i=2;
switch ( i )
        {
         case 1:  printf("valeur 1 "); break;
         case 2 :  printf("valeur 2 ");break;
        default   :   printf("valeur defaut ");
        }
```

Affichage ➔ valeur 2

Langage C : séance 2

Les instructions itératives :

- **L'instruction « while » :**

syntaxe :

Parenthèses obligatoires

```
while (expression_condition)
        instruction;
```

ou

```
while (expression_condition)
{
        instruction1;
        …
        instructionn;
}
```

- **L'instruction « do … while »** :

syntaxe :

```
do   {
        instruction ou bloc
} while (expression_condition);
```

Langage C : séance 2

Les instructions itératives :

- **L'instruction « for » :**

syntaxe : **for (**expression1 **;** expression2 **;** expression3**)**
instruction ou bloc instructions

signification :
- expression1 : initialisation de la variable compteur.
- expression2 : test donnant la condition de poursuite des itérations.
- expression3 : modification de la variable compteur.

exemple :

```
int tab[10];
int i;
for (i=0; i<10; i++)
        tab[i]=0;
```

```
char ch[10];
int i;
for (i=0;i<10;i++)
        ch[i]='\0';
```

L'instruction « break » dans une boucle permet d'en sortir.
L'instruction « continue » dans une boucle permet de passer à l'itération suivante.

Langage C : séance 2

Les opérateurs de base :

- **Les opérateurs d'affectation composés :**

Pour les opérateurs : +, -, *, /, %, &, |, ^, <<, >>
on peut remplacer :
x = x **opérateur** y; par x **operateur** = y;

Exemple : x**+=** y; équivalent à x=x**+**y;

- **Les opérateurs d'incrémentation :**

surincrémentation : (operande**++**) (idem avec operande**--**)
Exemple : y= a* (b++);
équivalent à : y=a*b;
b=b+1;

préincrémentation : (**++**opérande) (idem avec **--**operande)
Exemple : y=a*(++b);
équivalent à : b=b+1;
y=a*b;

Langage C : séance 2

Les opérateurs de base :

- **L'opérateur conditionnel :**

  Syntaxe : **expression1 ? expression2 : expression3**

  Signification :

  si (expression1 !=0) **alors** expression2 est évaluée
  sinon expression3 est évaluée.

  Exemple : (a!=0) ? (b=a):(b=c)

- **L'opérateur virgule :**

  Syntaxe : expression1**,** expression2

  Signification : L'expression 1 est évaluée et sa valeur oubliée, l'expression 2 est évaluée et sa valeur correspond à la valeur de l'ensemble.

  Exemple : x= (i=1, k=3); *// x vaudra 3, i vaudra 1 et k vaudra 3 après exécution*

L'opérateur de taille :

Syntaxe : **sizeof** nom_type ou **sizeof** nom_variable

Signification : retourne la taille en octets de nom_type.

Exemple : char ch[10]; ➔ sizeof ch retourne 10
int i; ➔ sizeof i retourne 4 (nbre d'octets occupé par un int en mémoire, dépend du SE).