

M. BERTHOMIER ERIC

INITIATION AU LANGAGE C

NIVEAU 1

APPRENTISSAGE DE LA SYNTAXE DE BASE

Version 1.1 du 13 Mars 2000

1 Premiers Pas

1.1 Prologue

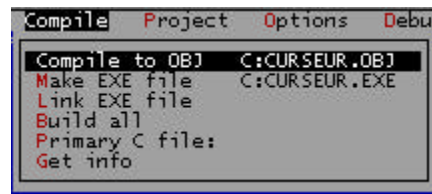
Ce cours utilise comme compilateur le Borland C 2.0. Celui-ci est disponible sur le Web.

1.2 Exemple de programme

```
main ()
{
    puts ("Bonjour");
}
```

Voici pour exemple un premier programme qui fonctionne malgré le fait qu'il ne soit pas normalisé. Celui-ci affiche le mot bonjour à l'écran.

Une fois le texte du programme frappé, il faut le compiler (Compile), c'est à dire en analyser la syntaxe.



On remarquera plusieurs types de commandes dans le menu Compile :

1. *Compile* qui compile le programme c'est à dire en analyse la syntaxe et produit un pseudo-code non interprétable par l'ordinateur (.obj)
2. *Make* qui construit un exécutable .exe qui permet au programme d'être exécuter comme n'importe quel autre programme.
3. *Link* qui permet de lier plusieurs pseudo-codes entre eux et d'en créer un
4. *Build all* qui permet de retraduire tous les codes source d'un projet (ensemble de programmes liés les uns aux autres).

Code source : le code source représente le programme sous sa forme de langage C, c'est à dire ce que vous tapez dans l'éditeur de texte du compilateur.

Pour notre part, nous n'utiliserons que Compile, le reste se faisant durant la exe lors de l'exécution de la commande Run.



1.3 Exécution du programme (Run)

On remarque que :

- └ Pour voir ce qu'affiche le programme il est nécessaire d'utiliser la commande UserScreen dans le menu Windows.
- └ L'exécution du programme de nombreuses fois fait apparaître le mot Bonjour plusieurs fois.

1.4 Correction du programme

Nous allons normaliser le programme. En fait, à sa base, le langage C n'est qu'un ensemble de bibliothèques à partir desquelles le compilateur trouve les fonctions et les applications qui lui permettent de créer un programme exécutable. Exactement ce que vous faites lorsque vous recherchez dans une encyclopédie.

Certaines bibliothèques sont incluses dans des compilateurs ce qui permet à notre programme de s'exécuter. Normalement, **puts** a besoin de la bibliothèque **stdio.h**. Pour ajouter une bibliothèque, il suffit d'ajouter **#include <nom de la bibliothèque>** en début de programme.

Le second point à corriger est l'absence de valeur de retour. La valeur de retour permet à un programme ou à l'utilisateur de savoir si le programme que l'on exécute s'est correctement terminé. 0 signifie une terminaison sans erreur.

En lui rajoutant quelques fonctionnalités on obtient donc :

```
#include <stdio.h>
#include <conio.h>

int main ()
{
    clrscr ();    /* Efface l'écran */
    puts ("Bonjour");
    getch ();    /* Attendre */
    return (0);
}
```

Bibliothèques

Corps du programme

La valeur de retour n'est pas obligatoire, pour ne pas utiliser de valeur de retour on utilise **void main ()** à la place de **int main ()**. **void** peut se traduire par "ne contenant rien".

Attention: dans ce cas, on utilise **return;** et non **return (0)**.

Le programme devient donc :

```
#include <stdio.h>
#include <conio.h>

void main ()
{
    clrscr ();    /* Efface l'écran */
    puts ("Bonjour");
    getch ();    /* Attendre */
    return;      /* Facultatif car c'est la dernière ligne (fin)*/
}
```

1.5 Petit mot sur ce qu'est une bibliothèque

A l'instar de l'étudiant qui recherche dans des livres, on peut dire que le ".h" représente l'index du livre et le ".c" le contenu du chapitre concerné, le ".o" ou ".

Exemple

Lorsque le compilateur C rencontre le mot `clrscr`, il regarde dans chacun des ".h" déclaré par l'instruction `#include` si ce mot y est défini. Il trouve celui-ci dans la `conio.h` et remplace donc ce mot par le code qui lui est associé au moment de la compilation. A l'inverse, s'il ne le trouve pas, celui-ci émettra une erreur de syntaxe.

1.6 Un exemple de fichier bibliothèque

Vous trouverez ci-dessous, un extrait de la bibliothèque `stdio.h`. On y retrouve notamment la définition de **puts** que l'on voit dans ce cours et la définition de **printf** que l'on verra dans le 2nd cours.

Extrait du fichier `stdio.h`

```
/*  stdio.h

Definitions for stream input/output.

Copyright (c) Borland International 1987,1988
All Rights Reserved.
*/
#if !defined(__STDIO_DEF_)
#define __STDIO_DEF_
int  _Cdecl printf (const char *format, ...);
int  _Cdecl puts  (const char *s);
#endif
```

1.7 Les différentes fonctions

puts : permet d'afficher du texte.

clrscr : permet d'effacer l'écran.

getch : permet d'attendre la frappe d'une touche.

/* Commentaire */ : permet de mettre un commentaire.

Notre programme efface l'écran puis affiche bonjour et attend que l'on appuie sur une touche afin que l'on puisse voir ce qu'il a écrit.

1.8 Squelette de programme

On peut définir le squelette d'un programme C de la façon suivante :

```
/* Déclaration des bibliothèques */

int main ()
{
```

```
/* Déclaration des variables */      cf. chapitre 2

/* Corps du programme */
getch ();      /* Facultatif mais permet de voir ce qui s'est produit à l'écran */
               /* En attendant l'appui d'une touche */
return (0);   /* Aucune erreur renvoyée */
}
```

1.9 Les blocs

La partie de programme située entre deux accolades est appelée un bloc. Je conseille de prendre l'habitude de faire une tabulation après le retour à la ligne qui suit l'accolade. Puis retirer cette tabulation après l'accolade fermante du bloc. Ainsi, on obtient :

```
{
    Tabulation
    Tout le code est frappé à cette hauteur
}
```

} Bloc de programme

Retrait de la tabulation
Tout le texte est maintenant frappé à cette hauteur.

Cette méthode permet de contrôler la fermeture de toutes les accolades et leurs correspondances.

1.10 Les commentaires

Commenter signifie qu'une personne ne connaissant pas le langage C doit pouvoir lire le programme et le comprendre.

Les commentaires sont indispensables dans tout bon programme.

Les commentaires peuvent être placés à n'importe quel endroit dans le programme. Ils commencent par /* et se terminent par */.

```
/* Commentaire */
```

1.11 Exercices d'application

Ecrire un programme qui écrit au revoir.

Ecrire un programme qui :

- ☞ Ecrit « Salut toi, appuie sur une touche s'il te plaît
- ☞ Attend l'appui d'une touche
- ☞ Efface l'écran
- ☞ Ecrit « Merci d'avoir appuyé sur une touche

Commentez le précédent programme.

Exemple :

```
puts ("Cours de programmation" );
/* Ecrit 'Cours de programmation' à l'écran */
```

Ecrire un programme qui écrit

« Hamlet says To be or not to be, that is the question. »

Corrigés des exercices du chapitre 1

! **Ecrire un programme qui écrit au revoir**

```
#include <stdio.h>
#include <conio.h>

main()
{
    clrscr();          /* Efface l'écran */
    puts("Au revoir "); /* Affiche Au revoir */
    getch ();
}
```

! **Ecrire un programme qui :**

- ☞ Ecrit « Salut toi, appuie sur une touche s'il te plaît
- ☞ Attend l'appui d'une touche
- ☞ Efface l'écran
- ☞ Ecrit « Merci d'avoir appuyé sur une touche

```
#include <stdio.h>
#include <conio.h>

int main ()
{
    clrscr (); /* Efface l'écran */

    puts ("Salut toi, appuie sur une touche s'il te plaît");
    /* Affiche le message Salut toi, ... s'il te plaît */

    getch (); /* Attend la frappe d'une touche */

    puts ("Merci d'avoir appuyé sur une touche");
    /* Affiche le message Merci d'avoir appuyé sur une touche */

    return 0; /* Revient */
}
```

! **Commentez le précédent programme.**

! **Ecrire un programme qui :**

Ecrit : « Hamlet says To be or not to be, that is the question. »

```
#include <stdio.h>
#include <conio.h>
int main()
{
    clrscr ();
    puts ("Hamlet says To be or not to be, that is the question.");
    getch ();
    return 0;
}
```

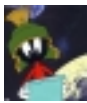
2 Les variables (1^{ère} partie)

2.1 Printf : fonction indispensable pour afficher une variable

Exemple :

```
#include <stdio.h>

int main ()
{
    printf ("Coucou c'est moi"); /* Affiche Coucou c'est moi à l'écran */
    getch ();                    /* Attendre l'appui d'une touche */
}
```



On pourrait dire que la fonction **printf** est la même que l'instruction **puts** vu précédemment mais il n'en est rien ... Celle-ci est beaucoup, beaucoup, beaucoup plus puissante.

Syntaxe:

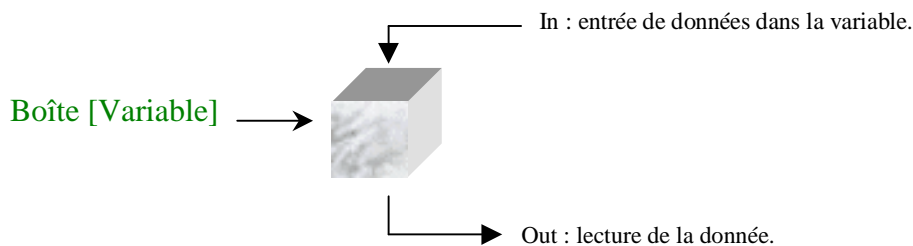
La syntaxe de printf est très complexe et pourrait à elle seule faire l'objet d'un cours, nous en verrons donc des applications au fur et à mesure des besoins.

2.2 Variable

Comme son nom l'indique une variable est quelque chose qui varie.
C'est vrai mais ce n'est pas suffisant.

Une variable peut être considérée comme une boîte dans laquelle on met des données que l'on peut lire ou écrire.

Variable : Lire ou Ecrire



La manière la plus facile de lire le contenu d'une variable est la fonction **printf** que l'on a aperçu précédemment.

La manière la plus simple de donner une valeur à une variable est l'opérateur mathématique **=**. Ecrire dans une variable ayant déjà une valeur revient à la modifier.

Une variable ne peut contenir qu'une seule chose à la fois. Si vous mettez une seconde donnée dans la variable, la précédente est effacée.

2.3 Déclaration d'une variable

La déclaration d'une variable se fait simplement en écrivant :

```
<son type> <son nom>;
```

Exemples de type de variables :

char : caractère

int: entier

2.4 Application : exemple

```
#include <stdio.h>
```

```
int main ()
```

```
{
  int    i;      /* i : variable de type entier */
  char   car;   /* car: variable de type caractère */
}
```

} Déclarations des variables

```
i = 65;        /* i vaut 65 */
car = 'E';     /* car vaut E */
```

} Ecriture dans les variables

```
clrscr ();    /* Efface l'écran */
```

```
printf ("i vaut %d.\n", i);    /* Affiche la valeur de i */
printf ("car vaut %c.\n",car); /* Affiche la valeur de car */
```

} Affichage des variables

```
getch ();    /* Attendre l'appui d'une touche */
```

```
return (0);
```

```
}
```

Explications :

On met dans la variable i la valeur 65.

On met dans la variable car la valeur de E.

Note : En informatique, tout n'est que nombre, je dis donc la valeur de E et non E car c'est le code Ascii de E qui est sauvegardé dans cette variable. Nous reviendrons là dessus un peu plus tard.

☞ printf ("i vaut **%d**.\n", i);
%d signifie que l'on attend une **valeur décimale**, le programme va donc remplacer le %d par la valeur de i.

☞ printf ("car vaut **%c**.\n", car);
%c signifie que l'on attend une **valeur de type caractère**, le programme va donc remplacer le %c par la valeur de car.

☞ \n permet de réaliser un retour chariot c'est à dire un retour à la ligne.

2.5 Utilisation multiple du %

Le code "%x" signifie que le compilateur C doit **remplacer** ce code par la valeur correspondante (qui lui est fourni dans la suite de l'instruction) en la **transformant** dans le type x. Cette transformation est appelé un **cast**.

Exemple :

```
int i;  
i =65;  
printf ("Le caractère %d est %c",i,i);
```

nous donnera l'affichage suivant :

Le caractère 65 est A.

Le %d est remplacé par la valeur numérique de i c'est à dire 65.

Le %c est remplacé par la valeur alphanumérique (ASCII) de i c'est à dire A.

cf. Table Ascii en Annexe.

2.6 Exercices d'applications directes

En utilisant ce qui a été fait précédemment, faites afficher les valeurs 70, 82, 185 et 30.

En utilisant ce qui a été fait précédemment, faites afficher, les caractères c, o, u, C, O, U.

2.7 Réutilisation d'une variable

On peut réutiliser une variable autant de fois que l'on veut, la précédente valeur étant effacée.

`i = 3; i = 5; i = 7;` donnera au final pour valeur de i la valeur de 7.

`car = 'E'; car = 'G'; car = 'h';` donnera au final pour valeur de car la valeur de 'h'.

2.8 Caractères spéciaux

Certains caractères utilisés par la fonction printf ("% " par exemple) ou même tout simplement pour déclarer une variable (' pour les caractères par exemple) oblige à utiliser le caractère de suffixe \ pour pouvoir être affiché.

Exemple :

☞ Pour afficher un % avec printf j'écrirai :

```
printf "La réduction était de 20 \%"
```

☞ Pour déclarer un caractère avec la valeur ' (prononcée cote en informatique et non pas apostrophe (français)), on écrira :

```
char car;  
car = \"
```

2.9 Exercices à réaliser

Faire un programme qui réalise l'affichage suivant en utilisant les variables

Aide : Sans retour chariot, on affiche à la suite

```
Coucou  
17
```

De la même façon, réaliser un programme qui réalise l'affichage suivant :

```
C  
O  
U
```

Cou

1

2

3

456

C'est rigolo ...

Rappel : pour mettre une variable `c` égale à `'` on écrit `c = '\''`

Ecrire un programme qui écrit

« Hamlet says "To be or not to be, that is the question." »

avec les " bien sûr.

Rappel : Pour pouvoir afficher un caractère de syntaxe C, par exemple `"`, on utilise le caractère `\` comme préfixe à ce caractère. Pour obtenir un `"`, on utilise donc `\`.

Corrigés des exercices du chapitre 2

! **Faites afficher, en utilisant ce qui a été fait précédemment, les valeurs 70, 82, 185 et 30.**

```
#include <stdio.h>

int main ()
{
    int i,a,b,c;
    i=70;
    a=82;
    b=185;
    c=30;

    clrscr ();
    printf ("i vaut %d.\n",i);
    printf ("a vaut %d.\n",a);
    printf ("b vaut %d.\n",b);
    printf ("c vaut %d.\n",c);
    getch ();

    return 0;
}
```

! **Faites afficher, en utilisant ce qui a été fait précédemment, les caractères c, o, u, C, O, U.**

```
#include <stdio.h>

int main ()
{
    char a,b,c,d,e,f;
    a='c';
    b='o';
    c='u';
    d='C';
    e='O';
    f='U';
    clrscr ();

    printf ("a vaut %c.\n",a);
    printf ("b vaut %c.\n",b);
    printf ("c vaut %c.\n",c);
    printf ("d vaut %c.\n",d);
    printf ("e vaut %c.\n",e);
    printf ("f vaut %c.\n",f);

    printf ("a, b, c, d, e, f
valent : %c, %c, %c, %c, %c,
%c.\n",a,b,c,d,e,f);

    getch ();
    return (0);
}
```

! **Faire un programme qui réalise l'affichage ...**

Coucou

17

```
#include <stdio.h>

int main ()
{
    char car = 'C';
    int  nbre = 17;

    printf ("%c",car);
    car = 'o';
    printf ("%c",car);
    car = 'u';
    printf ("%c",car);
    car = 'c';
    printf ("%c",car);
    car = 'o';
    printf ("%c",car);

    car = 'u';
    printf ("%c",car);

    printf ("\n%d",nbre);

    /* Attente */
    getch ();

    return (0);
}
```

! Faire un programme qui réalise l'affichage ...C
O
U

```

...
#include <stdio.h>

int main ()
{
    char car = 'C';
    int  nbre = 1;

    clrscr (); /* Efface l'écran
*/
    car = 'C';
    printf ("\n%c",car);
    car = 'O';
    printf ("\n%c",car);
    car = 'U';
    printf ("\n%c",car);
    car = 'C';
    printf ("\n%c",car);
    car = 'o';
    printf ("%c",car);
    car = 'u';
    printf ("%c",car);

    printf ("\n%d",nbre);
    nbre = 2;
    printf ("\n%d",nbre);
    nbre = 3;
    printf ("\n%d",nbre);
    nbre = 456;
    printf ("\n%d",nbre);
}

car = 'C';
printf ("\n%c",car);
car = '\';
printf ("%c",car);
car = 'e';
printf ("%c",car);
car = 's';
printf ("%c",car);
car = 't';
printf ("%c",car);
car = ' ';
printf ("%c",car);
car = 'r';
printf ("%c",car);
car = 'i';
printf ("%c",car);
car = 'g';
printf ("%c",car);
car = 'o';
printf ("%c",car);
car = 'l';
printf ("%c",car);
car = 'o';
printf ("%c",car);
/* Attente */
getch ();

return (0);

```

! Faire un programme qui écrit "Hamlet says ... "

```

#include <stdio.h>

int main ()
{
    clrscr (); /* Efface l'écran */

    printf ("Hamlet says : \"To be or no to be, that is the
question.\");

    /* Attente */
    getch ();

    return (0);
}

```

3 Les variables (2^{nde} partie)

Durant tout ce chapitre, nous aurons pour but simple, mais complet et complexe dans sa programmation, de faire une malheureuse calculatrice.

3.1 Exercice de mise en bouche

Ecrire un programme qui :

- Ecrit "Calculatrice :" **et** saute 2 lignes.
- Ecrit "Valeur de a :" **et** saute 1 ligne.
- Attend l'appui d'une touche.
- Ecrit "Valeur de b :" **et** saute 1 ligne.
- Attend l'appui d'une touche
- Ecrit "Valeur de a + b :"

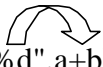
Pas à pas, nous allons maintenant réaliser notre petit programme de calculatrice.

3.2 Déclaration des variables

Complétez le programme en :

- Déclarant 2 variables a et b de type int (entier).
- Assignant à ces deux variables les valeurs de 36 et 54.
- Faisant afficher le résultat de la somme de a + b (et non pas afficher 90 !).

Pour faire afficher le résultat il est possible d'utiliser la fonction printf directement ou indirectement en utilisant une troisième variable. Pour plus de facilité, nous allons utiliser un affichage direct. Celui-ci s'effectue de la façon suivante :

N.B.  printf ("Valeur de a + b : %d",a+b);
Le %d est remplacé par la valeur de a+b

3.3 Saisie des variables



Si une calculatrice se limitait à exécuter la somme de deux nombres fixes, le boulier serait encore de mise ...

Pour saisir une variable, il est nécessaire d'utiliser la fonction scanf. La fonction scanf s'utilise de la façon suivante :

Saisie de la valeur a : `scanf ("%d", &a);`

Comme pour printf, on reconnaît le %d pour la saisie d'un nombre décimal. Le & devant le a signifie que l'on va écrire dans la variable a.



En fait &a signifie l'adresse mémoire de a. Scanf va donc écrire dans l'emplacement mémoire (la petite boîte contenant la variable) de a. Si l'on oublie le &, on écrira chez quelqu'un d'autre, à une autre adresse. Et là ça peut faire mal, très mal ...
Mais ceci est une autre histoire ...

Nous allons maintenant saisir les variables a et b. Pour exemple, je mets ici le code pour la saisie de a, la saisie de b reste à faire par vos soins à titre d'exercice.

```
/* Saisie de la valeur de a */  
printf ("Valeur de a :\n");  
scanf ("%d",&a);
```

Il est conseillé d'initialiser les variables avant de les utiliser, de ce fait, au début du programme, après leur déclaration, assignez à a et à b la valeur de 0.

Exemple :

```
a = 0;
```

Si tout c'est bien passé, vous avez maintenant entre les mains une super calculatrice

N.B. Appuyez sur la touche Enter après avoir taper votre valeur.

3.4 Déclaration avec initialisation d'une variable

Pour aller plus rapidement, il est possible d'initialiser une variable dans le même temps que sa déclaration. Pour cela, rien de plus simple, ajouter à la fin de la déclaration = suivi de la valeur d'initialisation.

Exemple :

```
int i = 10;
```

3.5 Evolution du logiciel : exercice

Aïe Aïe Aïe Dur Dur d'être informaticien. J'ai envie de faire une super calculatrice et je me dis qu'en fait la simplicité qui a eu lieu tout à l'heure n'est pas forcément adapté à

- Assigner une troisième valeur de type int (penser à l'initialiser à 0) que l'on nommera bêtement s comme somme.
- Une fois les valeurs de a et b saisies, initialisez s avec la valeur de a + b. Comme en mathématiques élémentaires.
- Affichez la valeur de s. On devrait avoir les même résultats qu'auparavant, off course.

3.6 Exercices d'application

Réalisez 2 petits programmes qui font :

- la soustraction de 2 nombres
- la multiplication de 2 nombres

3.7 Un nouveau type: les nombres à virgule ou flottants (float)

Le type **float** permet de déclarer un nombre à virgule. Transformez les 3 précédents programmes en utilisant le type float au lieu du type int et %f pour saisir les valeurs de a et b, et pour afficher le résultat.

Petite aide :

```
float a;  
printf ("Saisie de a :");  
scanf ("%f",&a);  
printf ("\na vaut : %f",a);  
getch ();
```

Ce petit morceau de programme saisit la valeur de a et la fait afficher.

Pour un affichage plus agréable il est possible de fixer le nombre de chiffres après la virgule de la façon suivante **%.[nombre de chiffres après la virgule]f**.

Exemple :

```
printf ("% .2f",a);
```

3.8 **Ouah, les 4 opérations !!!**

Créer un quatrième programme qui réalise la division de deux nombres.

Tester avec une division par 0 !!!

La solution à ce petit problème sera vu dans le cours des conditions (if).

3.9 **Exercices à réaliser**

Compléter les 4 programmes afin de réaliser les opérations suivantes :

$$\left\{ \begin{array}{l} s = a + b + c \\ s = a - b - c \\ s = a / b / c \\ s = a * b * c \end{array} \right.$$

où a,b,c sont des nombres à virgules (float).

La fonction **abs** permet d'obtenir la valeur absolue d'un nombre entier. Utiliser cette fonction pour calculer la valeur absolue de (a-b).

Exemple d'utilisation de abs :

```
S = abs (a-b);
```

La fonction **ceil** permet d'obtenir l'arrondi entier supérieur d'un nombre réel. Utiliser cette fonction pour calculer l'arrondi supérieur de (a / b).

Exemple d'utilisation de ceil :

```
S = ceil (a/b);
```

Corrigés des exercices du chapitre 3

! **Soustraction de 2 nombres**

```
#include <stdio.h>
#include <conio.h>

int main ()
{
    int a,b;           /* Déclaration des variables */
    int d;             /* Différence entre les 2 nombres */

    a=0;              /* Initialisation des variables */
    b=0;

    clrscr ();
    printf("Calculatrice :\n\n");
    printf("Valeur de a : ");
    scanf("%d",&a);
    printf("\n");
    printf("Valeur de b : ");
    scanf("%d",&b);

    d=a-b;
    printf("Valeur de a-b : %d\n",d); /* Affichage de la différence */

    getch ();
    return 0;
}
```

! **Multiplication de 2 nombres**

```
#include <stdio.h>
#include <conio.h>

int main ()
{
    int a,b;           /* Déclaration des variables */
    int m;             /* Résultat de la multiplication */
    a=0;              /* Initialisation des variables */
    b=0;

    clrscr ();
    printf("Calculatrice :\n\n");
    printf("Valeur de a : ");
    scanf("%d",&a);
    printf("\n");
    printf("Valeur de b : ");
    scanf("%d",&b);

    m = a*b;
    printf("Valeur de a*b : %d\n", m);
    /* Affichage de la multiplication */

    getch ();
    return 0;
}
```


! **Transformez les 3 précédents programmes avec le type float**

Aucune difficulté dans cet exercice, je ne mettrai ici la correction que de la multiplication, les autres opérations se réalisent sur le même schéma.

```
int main ()
{
    float a,b;           /* Déclaration des variables */
    float m;            /* Résultat de la multiplication */

    a=0;                /* Initialisation des variables */
    b=0;

    clrscr ();
    printf("Calculatrice :\n\n");
    printf("Valeur de a : ");
    scanf("%f",&a);
    printf("\n");
    printf("Valeur de b : ");
    scanf("%f",&b);

    m = a*b;
    printf("Valeur de a*b : %f\n", m);
    /* Affichage de la multiplication */

    getch ();
    return 0;
}
```

Pour l'addition :

```
m = a + b;
printf ("Valeur de a+b : %f", m);
```

Pour la soustraction :

```
m = a - b;
printf ("Valeur de a-b : %f", m);
```

! **Calculer la somme a + b + c**

```
#include <stdio.h>
#include <conio.h>
```

```
int main ()
{
    float a,b,c,s;      /* Déclaration des variables */

    a=0;                /* Initialisation des variables */
    b=0;
    c=0;
    s=0;

    clrscr ();
    printf("Saisie de a : ");
    scanf("%f",&a);
    printf("Saisie de b : ");
    scanf("%f",&b);
    printf("Saisie de c : ");
    scanf("%f",&c);
} /* Saisie variables flottantes */
```

```
s = a+b+c; /* Calcul de la somme */
printf("Valeur de s : %.2f\n",s); /* Affichage de la somme */

getch ();
return 0;
}
```

! **Calculer la différence $a - b - c$**

```
d = a-b-c; /* Calcul de la différence */
printf("Valeur de d : %.2f\n",d); /* Affichage de la différence */
```

! **Calculer la multiplication $a * b * c$**

```
m = a*b*c; /* Calcul de la multiplication */
printf("Valeur de m : %.2f\n",m); /* Affiche la multiplication */
```

! **Calculer la division $a / b / c$**

```
d = a/b/c; /* Calcul de la division */
printf("Valeur de d : %.2f\n",d); /* Affichage de la division */
```

! **Calculer la valeur absolue de $a - b$**

```
r=abs(a-b); /* Calcul de la valeur absolue */
printf("Valeur de r : %f\n",r); /* Affiche la valeur absolue */
```

! **Calculer l'arrondi de $a + b$**

```
c=ceil(a/b); /* Calcul de l'arrondi */
printf("Valeur de c : %f\n",c); /* Affichage de l'arrondi */
```

N Il aurait été possible d'utiliser **%d** du fait que l'arrondi est un nombre entier !

4 Conditions

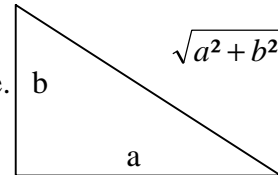
Avec des Si on mettrait Paris en bouteille...

4.1 Exercice de mise en bouche

Ecrire un programme qui met en application le théorème de Pythagore pour calculer l'hypoténuse d'un triangle rectangle.

Rappel :

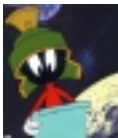
Dans un triangle rectangle, les longueurs côtés sont définies comme le montre la figure ci-contre.



Note :

- ≡ La racine carrée s'obtient par l'utilisation de la fonction **sqrt** (valeur) contenue dans la bibliothèque math.h. (`#include <math.h>`)
- ≡ a^2 peut s'obtenir par `a*a`.

Méthodologie :



1. Rechercher les variables nécessaires et les déclarer dans le programme.
2. Faire saisir a au clavier.
3. Faire saisir b au clavier.
4. Effectuer l'opération de la racine carrée et afficher le résultat.

4.2 Les conditions : Si Alors Sinon

if (condition vraie)	si (condition vraie)
{	{
instructions 1	alors
}	faire instructions 1
else	}
{	sinon
instructions 2	{
}	faire instructions 2
	}

Les conditions s'expriment avec des opérateurs logiques ...

4.2.1 Opérateurs logiques relationnels

Ils servent à comparer deux nombres entre eux.

Libellé	Opérateur
<i>Inférieur</i>	<
<i>Supérieur</i>	>
<i>Equivalent</i>	==
<i>Différent</i>	!=
<i>Inférieur ou égal</i>	<=
<i>Supérieur ou égal</i>	>=

4.2.2 Opérateurs logiques purs

Ce sont des opérateurs logiques permettant de combiner des expressions logiques.

<i>Libellé</i>	<i>Opérateur</i>
<i>Et (and)</i>	&&
<i>Ou (or)</i>	
<i>Non (not)</i>	!

N.B. | se nomme en informatique un pipe.

4.2.3 Vrai ou faux

La valeur Vrai peut être assimilée à la valeur numérique 1 ou à toute valeur > 0.

La valeur Faux peut être assimilée à la valeur numérique 0.

L'opérateur Ou (||) correspond alors à une addition

Ou	Vrai	Faux	+	1	0
Vrai	Vrai	Vrai	1	2	1
Faux	Vrai	Faux	0	1	0

L'opérateur Et (&&) correspond alors à une multiplication

Et	Vrai	Faux	*	1	0
Vrai	Vrai	Faux	1	1	0
Faux	Faux	Faux	0	0	0

On notera que !Vrai = Faux et !Faux = Vrai.

4.2.4 Combinaison

Toutes les opérations logiques peuvent se combiner entre elles. La seule condition d'utilisation d'un si (if) avec de telles combinaisons est de l'entourer de ().

Exemple :

```
if ((car == 'a') || (car == 'A'))
```

4.2.5 Astuce

Vous verrez souvent ce type de code écrit :

```
if (er)
{
    /* Alors faire quelque chose */
}
```

En appliquant ce qui a été vu précédemment, on en déduit que ce code signifie que

```
si (er != 0) /* si er différent de 0 */
{
    /* Alors faire quelque chose */
}
```

4.3 Les accolades

Les accolades entourant les blocs d'instructions d'une condition peuvent être omises si le bloc est constitué d'une seule instruction.

Exemple :

```
if (car == 'b')
    printf ("car vaut b.");
else
    printf ("car est différent de b.");
```

4.4 Exercices

! Faites saisir une variable de type entière et indiquez à l'utilisateur si celle-ci est positive ou négative ou nulle.

Aide :

```
if (a>0)
{
    printf ("Valeur positive");
}
else
{
    printf ("Valeur négative");
}
```

! Faites saisir une variable de type entière et indiquez à l'utilisateur si celle-ci est positive, négative ou nulle.

! Faites saisir une variable de type caractère et indiquez à l'utilisateur si celle-ci est une voyelle.

Corrections des exercices du chapitre 4

! *Ecrire un programme qui effectue le théorème de Pythagore*

```
#include <stdio.h>
#include <conio.h>
#include <math.h>

int main ()
{
    float a;    /* base du triangle */
    float b;    /* côté du triangle rectangle */
    float p;    /* valeur de l'hypoténuse (p pour Pythagore !) */

    /* Initialisation des variables pour palier aux erreurs */
    a = 0;
    b = 0;

    /* Effacer l'écran */
    clrscr ();

    /* Saisie de a */
    printf ("Valeur de la base : ");
    scanf ("%f",&a);

    /* Saisie de b */
    printf ("Valeur du côté : ");
    scanf ("%f",&b);

    /* Calcul par Pythagore */
    p = sqrt (a*a + b*b);

    /* Affichage du résultat */
    printf ("L'hypoténuse mesure : %.2f",p);

    /* Attendre avant de sortir */
    getch ();

    return (0);
}
```

! *Test du signe d'une valeur saisie au clavier*

```
#include <stdio.h>
#include <conio.h>
#include <math.h>

int main ()
{
    /* Valeur que l'on va saisir */
    int a = 0;

    /* Efface l'écran */
    clrscr ();

    /* Saisie de a */
    printf("Saisie de a : ");
```

```
scanf("%f",&a);

/* Test condition a<0 */
if (a<0)
{
    printf("la variable a est négative.\n");
}
else
{
    /* Test condition a>0 */
    if (a>0)
    {
        printf("la variable a est positive\n");
    }
    /* Sinon a est nulle */
    else
    {
        printf("la variable a est nulle\n");
    }
}

getch ();
return (0);
}
```

! **Test du signe d'une valeur saisie au clavier**

```
#include <stdio.h>
#include <conio.h>
#include <math.h>

int main ()
{
    /* Valeur que l'on va saisir */
    char car;

    /* Efface l'écran */
    clrscr ();

    /* Saisie du caractère a */
    printf("Saisie du caractère : ");
    scanf("%c",&car);

    /* Test condition car voyelle minuscule */
    if ((car == 'a') || (car == 'e') || (car == 'i') || (car == 'o') ||
        (car == 'u') || (car == 'y'))
    {
        printf("la variable car est une voyelle.\n");
    }
    else
    {
        printf("la variable car est une consonne.\n");
    }

    getch ();
    return (0);
}
```

5 Remise en forme

5.1 Prologue

L'objet de ce cours est de réaliser un petit break dans l'apprentissage du C et de s'attacher à voir que l'on est capable de réaliser avec le peu de moyen que l'on a.

Ce cours sera donc constitué de 3 exercices de difficultés croissantes avec apprentissage d'une nouvelle fonction et d'un exercice complet de programmation.

L'exercice 4 est assez difficile et mérite surtout de bien réfléchir avant de se mettre au clavier.

5.2 Exercice 1.

Réaliser un programme qui saisisse un nombre et indique à l'utilisateur si celui-ci est plus grand ou plus petit qu'un autre nombre fixé par le programme.

Exemple :

```
si (nbre_saisi<10)
alors "plus petit"
```

Reprendre l'exercice du chapitre 4 qui disait si un nombre est positif, négatif ou nul.

5.3 Retour sur Getch ()

La fonction `getch ()` permet d'attendre la frappe d'un caractère au clavier, de le lire et de le renvoyer. 2 utilisations peuvent être faites de `getch ()`, la première est celle permettant d'attendre la frappe d'une touche sans se soucier de sa valeur, la seconde est celle permettant de lire un caractère au clavier.

Exemples:

1. Attente
`getch ();`
2. Saisie d'un caractère
`char car;`
`car = getch ();`

A chaque fois, `getch ()` effectue le même traitement :

- Attend la frappe d'une touche au clavier.
- Renvoie le caractère frappé.

Dans le 1^{er} cas, ce caractère n'est simplement pas récupéré.

5.4 Boucle Faire ... Tant que (vrai)

Do ... while, traduisez par Faire Tant que permet de réaliser une suite d'événements tant qu'une condition ou un ensemble de conditions est rempli.

Exemple

```
char car;
```



```

int  sortie;

do
{
    clrscr ();
    printf ("Tapez S pour sortir ...");

    /* On saisit un caractère */
    car = getch ();

    /* On le compare pour savoir si l'on peut sortir */
    sortie = ((car == 's') || (car == 'S'));
} while (!sortie);

```

} Suite d'évènements

Rappel :

Un nombre entier vaut la valeur logique **vraie** si celui-ci est différent de 0.

Un nombre entier vaut la valeur logique **faux** si celui-ci est égal à 0.

|| signifie un ou logique (or).

5.5 Exercice 2

Tapez l'exemple précédent, aménagez le, comprenez le, puis transformez-le afin que l'on sorte de la boucle uniquement lorsque l'utilisateur a tapé le nombre 10.



Attention La saisie d'un nombre ne se fait pas par getch mais par scanf ...
Cf. Chapitre 3.

5.6 Exercice 3

Voici un petit exemple de programme qui permet d'obtenir des nombres aléatoires entre 0 et 100.

Notes :

random et **randomize** sont définis dans la bibliothèque <stdlib.h>.

```

#include <stdio.h>
#include <stdlib.h>

int main()
{
    int  nb_alea;    /* Nombre aléatoire */

    clrscr ();
    randomize ();

    /* Le nombre aléatoire est stocké dans une variable puis affiché */
    nb_alea = random (101);
    printf ("%d",nb_alea);

    /* Le nombre aléatoire est affiché directement à l'écran */
    printf ("%d",random (101));

    getch ();
}

```

randomize permet d'initialiser le système aléatoire.

random permet d'obtenir un nombre entre 0 et (n-1). [random (n)]

En vous aidant de ce petit programme et de ce qui a été fait précédemment, réaliser un petit jeu qui :

1. Initialise un nombre entre 0 et 100.
2. Tente de faire deviner ce nombre à l'utilisateur en lui indiquant s'il est plus petit ou plus grand.

Pour cet exercice, aidez vous de ce qui a été fait précédemment.

5.7 Exercice 4 : jeu de 421

Vous allez réaliser un mini jeu de 421. Les règles sont simples, vous avez 4 essais pour réaliser avec les 3 dés un jet donnant 421. Le programme lance les dés et affiche le résultat. Si c'est un 421, on affiche "Gagné", sinon on demande à l'utilisateur d'appuyer sur la touche R pour relancer. Au bout du 4^{ème} essai, au lieu de proposer de relancer, on affichera "Perdu" et le programme se terminera.

5.7.1 Analyse (ne pas toucher à l'ordinateur !).

Analyser l'énoncé du problème.

1. Identifier chaque étape du problème, ce qu'il faut faire pas à pas.
2. Analyser **chaque étape** en utilisant des mots *français* sans utiliser de terme informatique afin d'avoir un fonctionnement pas à pas ...

Faites comme si vous étiez un robot auquel on apprend comment jouer.

5.7.2 Programmation

Courage : Programmer le Jeu ...

Corrigés des exercices du chapitre 5

! §5.2

```
#include <stdio.h>

int main ()
{
    int    nb_choisi = 33;
    int    nb_saisi = 0;

    clrscr (); /* Efface l'écran */

    printf ("Votre nombre : ");
    scanf ("%d",&nb_saisi);

    if (nb_choisi < nb_saisi)
        printf ("Mon nombre est plus petit");
    else
    {
        if (nb_choisi == nb_saisi)
            printf ("Mon nombre est égal");
        else
            printf ("Mon nombre est plus grand");
    }

    /* Attente */
    getch ();

    return (0);
}
```

! §5.4

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>

int main ()
{
    int    valeur;

    do
    {
        clrscr ();
        printf ("Votre nombre : ");
        scanf ("%d",&valeur);
    }while (valeur != 10);
    return (0);
}
```

! §5.5

```
#include <stdio.h>
#include <stdlib.h> /* pour random */
#include <conio.h>

int main ()
```

```
{
    int  nb_hasard = 0;
    int  votre_nb = 0;

    randomize ();
    nb_hasard = random (101); /* Nombre entre 0 et 100 */

    do
    {
        clrscr ();

        printf("Saisie de votre nombre : ");
        scanf("%d",&votre_nb);

        if (nb_hasard < votre_nb)
        {
            printf ("\nMon nombre est plus petit");

            /* A cause du clrscr () qui nous empêcherait de voir le
            message */
            printf ("\nAppuyez sur une touche");
            getch ();
        }
        else
        {
            if (nb_hasard > votre_nb)
            {
                /* il peut être aussi égal ... */
                printf ("\nVotre nombre est plus grand");

                /* A cause du clrscr () qui nous empêcherait
                de voir le message */

                printf ("\nAppuyez sur une touche");
                getch ();
            }
        }
    }while (votre_nb != nb_hasard);

    printf ("\nTrouvé");
    getch ();

    return (0);
}
```

! **Le jeu du 421 : version longue**

```
#include <stdio.h>
#include <stdlib.h>

/* Note : true ou vrai est équivalent à la valeur 1 */
/* false ou faux est équivalent à la valeur 0 */
main()
{
    int  de1, de2, de3;
    int  gagne;
    int  essai;

    essai = 0;
    gagne = 0;
```

```
randomize ();

do
{
    clrscr ();

    de1 = random (6) + 1;
    de2 = random (6) + 1;
    de3 = random (6) + 1;

    printf ("Lancer : [%d] [%d] [%d]",de1,de2,de3);
    getch ();

    if (de1 == 4)
    {
        if (de2 == 2)
        {
            if (de3 == 1)
                gagne = 1;
            else
                gagne = 0;
        }
        else
        {
            if (de2 == 1)
            {
                if (de3 == 2)
                    gagne = 1;
                else
                    gagne = 0;
            }
            else
                gagne = 0;
        }
    }

    if (de2 == 4)
    {
        if (de1 == 2)
        {
            if (de3 == 1)
                gagne = 1;
            else
                gagne = 0;
        }
        else
        {
            if (de1 == 1)
            {
                if (de3 == 2)
                    gagne = 1;
                else
                    gagne = 0;
            }
            else
                gagne = 0;
        }
    }

    if (de3 == 4)
```

```
    {
        if (de2 == 2)
        {
            if (de1 == 1)
                gagne = 1;
            else
                gagne = 0;
        }
        else
        {
            if (de2 == 1)
            {
                if (de1 == 2)
                    gagne = 1;
                else
                    gagne = 0;
            }
            else
                gagne = 0;
        }
    }

    essai = essai + 1;
}
while ((!gagne) && (essai != 4));

if (gagne)
    printf ("\nGagné !!!");
else
    printf ("\nPerdu ...");

getch ();
}
```

! **Le jeu du 421 : version courte**

```
#include <stdio.h>
#include <stdlib.h>
```

```
main()
{
    int  de1, de2, de3;
    int  gagne;
    int  essai;

    essai = 0;

    randomize ();

    do
    {
        clrscr ();

        de1 = random (6) + 1;
        de2 = random (6) + 1;
        de3 = random (6) + 1;

        printf ("Lancer : [%d] [%d] [%d]", de1, de2, de3);
        getch ();

        gagne = ((de1 == 4) && (de2 == 2) && (de3 == 1));
    }
}
```

```
gagne = gagne || ((de1 == 4) && (de2 == 1) && (de3 == 2));
gagne = gagne || ((de1 == 2) && (de2 == 1) && (de3 == 4));
gagne = gagne || ((de1 == 2) && (de2 == 4) && (de3 == 1));
gagne = gagne || ((de1 == 1) && (de2 == 2) && (de3 == 4));
gagne = gagne || ((de1 == 1) && (de2 == 2) && (de3 == 4));

    essai = essai + 1;
}
while ((!gagne) && (essai != 4));

if (gagne)
    printf ("\nGagné !!!");
else
    printf ("\nPerdu ...");

getch ();
}
```

6 Et les shadoks pompaient ...

6.1 While

De la même façon que Do While, il est possible d'utiliser While (condition == Vrai) ...

Exemple:

```
char car;

while ((car != 's') && (car != 'S'))
{
    car = getch ();
}
```

6.2 Et les shadoks apprenaient que reprendre équivaut à apprendre ...

6.2.1 Exercice 1.1

Traduire en langage C, complétez avec les variables nécessaires, compilez, exécutez,

- Faire
- Effacez l'écran
- Saisir une touche
- Tant Que (touche != S) et (touche != s)

6.2.2 Exercice 1.2

Traduire en langage C, complétez avec les variables nécessaires, compilez, exécutez,

- Faire
- Effacez l'écran
- Saisir un nombre
- Tant Que (nombre != 10)

6.3 La fonction toupper ()

Le problème de la comparaison de la minuscule **et** de la majuscule peut être détourné par l'utilisation de la fonction toupper qui transforme un caractère minuscule en majuscule. La fonction toupper s'utilise de la façon suivante :

```
char car;
char car_min;

car_min = 'a'
car = toupper (car_min);
printf ("%c",car);
```

affichera A

6.4 O tant que en emporte le shadok ...

6.4.1 Exercice 2.1

Ecrire le programme...

Tant que je ne tape pas un nombre impair compris entre 1 et 10 je recommence la saisie d'un nombre.

6.4.2 Exercice 2.2

Ecrire le programme...

Tant que je ne tape pas une voyelle je recommence la saisie d'une touche.

6.5 Et les shadoks continuaient à pomper pour obtenir le résultat ...

Dans les exercices qui suivent, la notion de **compteur** intervient. Un compteur est une variable numérique que l'on décrémente (-1) ou incrémente (+1) suivant nos besoins.

Exemple :

```
int    i;
```

```
i++;   /* Incrémente le compteur défini par i */
```

```
i--;   /* Décrémente le compteur défini par i */
```

i++; revient à la même chose que i=i+1;

i--; revient à la même chose que i = i-1;

Le nombre de caractères peut donc être comptabilisé en ajoutant 1 à une variable à chaque fois qu'une touche est frappée.

6.5.1 Exercice 3.1

Ecrire le programme...

Tant que je n'ai pas saisi 10 caractères, je recommence la saisie d'une touche.

6.5.2 Exercice 3.2

Ecrire le programme...

Tant que je n'ai pas saisi 10 nombres, je recommence la saisie d'un nombre.

6.6 Au Parc des Shadoks, on trie, voyelles, chiffres premiers ...

6.6.1 Exercice 4.1

Ecrire le programme...

Tant que je n'ai pas saisi 10 voyelles, je recommence la saisie d'une touche.

6.6.2 Exercice 4.2

Ecrire le programme...

Tant que je n'ai pas saisi 10 chiffres premiers (1,3,5,7), je recommence la saisie d'un chiffre.

Corrigés des exercices du chapitre 6

! **Exercice 1.1**

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

int main ()
{
    char touche;

    do
    {
        clrscr ();
        a=getch ();
    }while((touche!='S')&&(touche!='s'));

    return (0);
}
```

! **Exercice 1.2**

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

int main ()
{
    int a = 0;

    do
    {
        clrscr ();
        printf("Saisie de a:");
        scanf ("%d",&a);
    }while (a!=10);
    return (0);
}
```

! **Exercice 2.1**

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

int main ()
{
    int a = 0;

    do
    {
        clrscr ();
        printf("Saisie de a:");
        scanf ("%d",&a);
    } while ( (a!=1) && (a!=3) && (a!=5) && (a!=7) && (a!=9));

    return (0);
}
```

! Exercice 2.2

Afin de faciliter la correction, j'utiliserai la fonction **toupper** qui permet de transformer un caractère minuscule en majuscule.

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

int main ()
{
    char a;

    do
    {
        clrscr ();
        printf("Tapez une lettre:");

        a=toupper (getch());

    }
    while((a!='A')&&(a!='E')&&(a!='I')&&(a!='O')&&(a!='U')&&(a!='Y'));
    return (0);
}
```

! Exercice 3.1

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

int main ()
{
    char car;
    int nb_frappes = 0; /* nombre de caractères frappés */

    do
    {
        clrscr ();
        car = getch (); /* 1 caractère frappé */
        nb_frappes ++;

    } while (nb_frappes != 10);

    return (0);
}
```

! Exercice 3.2

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

int main ()
{
    int nbre;
    int nb_frappes = 0; /* nombre de caractères frappés */

    do
    {
        clrscr ();
```

```
printf ("Votre nombre : ");
scanf ("%d",&nbre);          /* 1 nombre saisi */

nb_frappes ++;

} while (nb_frappes != 10);

return (0);
}
```

! Exercice 4.1

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

int main ()
{
    char a;
    int nb_voyelles = 0;

    do
    {
        clrscr ();
        printf("Tapez une lettre:");

        a=toupper (getch());
        if ((a=='A') || (a == 'E') || (a == 'I') || (a == 'O')
            || (a == 'U') || (a == 'Y'))
        {
            nb_voyelles ++;
        }
    } while (nb_voyelles != 10);

    return (0);
}
```

! Exercice 4.2

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

int main ()
{
    int nbre;
    int nb_nbre_premiers = 0;

    do
    {
        clrscr ();
        printf("Votre nombre : ");
        scanf ("%d",&nbre);

        if ((nbre == 1) || (nbre == 3) || (nbre == 5) || (nbre == 7))
        {
            nb_nbre_premiers ++;
        }
    } while (nb_nbre_premiers != 10);

    return (0);
}
```

}

7 Gestion de l'écran

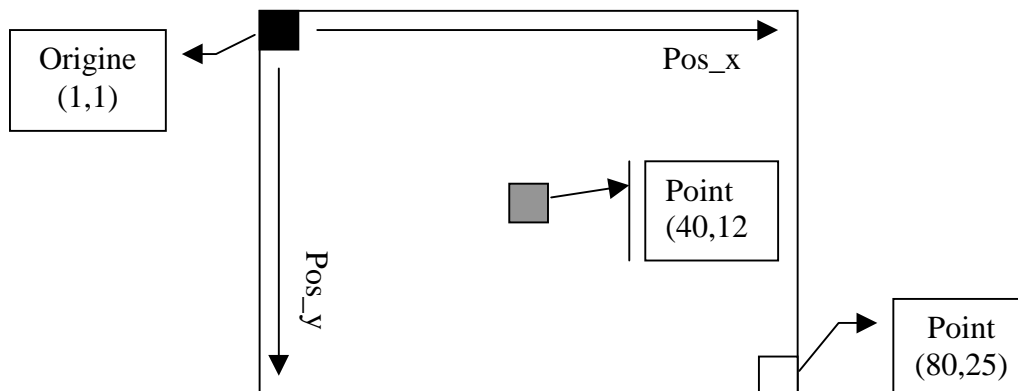
7.1 Prologue

Nous allons voir dans ce chapitre quelques fonctions permettant de se positionner et d'afficher à l'écran. Pour utiliser ces fonctions nous aurons besoin de la bibliothèque conio.h. (`#include <conio.h>`)

7.2 L'écran

7.2.1 Description

Un écran Dos se décompose de 80 colonnes et 25 lignes. Pour se positionner à un endroit précis de l'écran, on utilise la fonction **gotoxy (pos_x, pos_y)**; pos_x allant de 1 à 80 et pos_y allant de 1 à 25. L'origine est placée dans le coin haut gauche.



Pour faire afficher un caractère ou une chaîne de caractère à cet endroit, on utilise printf.

Exemple :

```
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>

int main ()
{
    clrscr ();
    gotoxy (70,25);
    printf ("Coucou");
    getch ();
    return (0);
}
```

7.2.2 Exercice d'application n°1

Faire afficher en (12,7) "Bonjour c'est moi et j'utilise le C" puis afficher en (23,18) "C'est bien on peut faire ce que l'on veut".

7.3 Choix multiple : **switch ... case**

7.3.1 Définition

Switch ... case permet l'exécution d'une série d'instructions dans le cas où une variable a une valeur précise.

On peut traduire **switch ... case** par dans le cas où la variable vaut ... faire ...

En plus des cas où, il est possible d'ajouter le cas **default** qui signifie par défaut faire ...

Exemple commenté

```
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>

int main ()
{
    char car;

    clrscr ();
    gotoxy (1,1);
    printf ("Saisie d'une touche : ");
    car = getch ();
    gotoxy (1,3);

    /* Réaliser les tests de cas sur la variable car */
    switch (car)
    {
        /* Dans le cas où car est égal au caractère a,e,i,o,u ou
        y afficher "Voyelle" */
        case 'a':
        case 'e':
        case 'i':
        case 'o':
        case 'u':
        case 'y':
            printf ("Voyelle\n");
        /* Fin des instructions concernant un caractère a,e,i,o,u
        ou y */
            break;

        /* Dans le cas où car est égal au caractère 1 afficher
        "Chiffre 1" */
        case '1':
            printf ("Chiffre 1\n");
            break;

        /* Dans les autres cas afficher "Rien à faire pour ce
        cas" */
        default:
            printf ("Rien à faire pour ce cas\n");
            break;
    }

    getch ();
    return (0);
}
```

7.3.2 Exercice d'application n°2

Réaliser un programme qui fait afficher :

- ☞ "Gauche" lorsque l'on appuie sur g ou G
- ☞ "Droite" lorsque l'on appuie sur d ou D
- ☞ "Haut" lorsque l'on appuie sur h ou H
- ☞ "Bas" lorsque l'on appuie sur b ou B

7.4 Opérateurs unaires

Les opérateurs unaires (unary) du C sont des raccourcis mathématiques permettant de réaliser des opérations mathématiques ou logiques simples utilisant 2 fois la même variable. Ils facilitent aussi les calculs de la machine.

Exemple :

`a = a + 5;` s'écrit `a += 5;`

Quelques opérateurs

<i>Opérateur</i>	<i>Signification</i>	<i>Exemple</i>
<i><code>a += val</code></i>	<code>a = a + val</code>	<code>a += 3</code>
<i><code>a -= val</code></i>	<code>a = a - val</code>	<code>a -= 7</code>
<i><code>a *= val</code></i>	<code>a = a * val</code>	<code>a *= 9</code>
<i><code>a /= val</code></i>	<code>a = a / val</code>	<code>a /= 11</code>

Il existe bien d'autres opérateurs unaires notamment logique mais ceci dépassant l'étendue de ce cours, je vous renvoie aux livres qui traite cela très bien.

7.5 Exercice : animation d'un curseur à l'écran

Traduire en langage C, compilez, exécutez, comprenez ...

Déclarer une variable x (initialisée à 10) de type entier.
 Déclarer une variable y (initialisée à 10) de type entier.
 Ces variables représenteront les positions du curseur à l'écran.

Déclarer une variable car de type caractère.
 Déclarer une variable sortie de type entier (vrai ou faux en fait) initialisée à faux (0).

Faire

Effacer l'écran.
 Afficher le caractère # en position x,y

Saisir un caractère sans écho et le mettre dans car.

Dans le cas où car est égal à :

G ou g : `x = x - 1 (x--)`
 D ou d : `x = x + 1 (x++)`
 H ou h : `y = y - 1 (y--)`
 B ou b : `y = y + 1 (y++)`
 S ou s : `sortie = vrai`
 Autre : réaliser un beep

Fin du Dans le cas où

Tant que (sortie = faux) /* ou !sortie */

Effacer l'écran

Aide :

Pour réaliser un beep, écrivez printf ("%c",0x07);

7.6 Complément d'exercice

Si l'on arrive au bord de l'écran ($x < 1$, $x > 80$, $y < 1$, $y > 25$), faire en sorte que le curseur passe de l'autre côté de l'écran.

Si $x < 1$ alors $x = 80$

Si $x > 80$ alors $x = 1$

Si $y < 1$ alors $y = 25$

Si $y > 25$ alors $y = 1$

7.7 La cerise sur le gâteau ...

Les touches de direction Haut Bas Droite Gauche sont en fait composées de 2 touches la touche de code Ascii 0 et respectivement les codes Ascii des touches H, P, K, M.

Pour pouvoir utiliser ces touches, il faut donc faire un double switch de la façon suivante :

```
switch (car)
{
    case 0:
        car = getch ();
        switch (car)
        {
            case 'H':
                y --;
                break;
            ...
        }
}
```

Complétez le programme pour qu'il fonctionne aussi avec les touches de direction.

Corrigés des exercices du Chapitre 7

! **Exercice d'application n°1**

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

int main ()
{
    clrscr ();
    gotoxy (12,7);
    printf ("Bonjour, c'est moi et j'utilise le C");
    gotoxy (23,18);
    printf ("C'est bien, on peut faire ce que l'on veut");
    getch ();
    return (0);
}
```

! **Exercice d'application n°2**

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

int main ()
{
    char car;

    clrscr ();
    gotoxy (1,1);
    printf ("Saisie d'une touche :");
    car = getch ();
    gotoxy (1,3);

    switch (car)
    {
        case 'g':
        case 'G':
            printf ("Gauche\n");
            break;

        case 'd':
        case 'D':
            printf ("Droite\n");
            break;

        case 'h':
        case 'H':
            printf ("Haut\n");
            break;

        case 'b':
        case 'B':
            printf ("Bas\n");
            break;

        default:
            printf ("Rien à faire pour ce cas\n");
            break;
    }
}
```

```
    }  
  
    getch ();  
    return (0);  
}
```

! Exercice : animation d'un curseur à l'écran

```
#include <stdio.h>  
#include <stdlib.h>  
#include <conio.h>  
  
int main ()  
{  
    char car;  
    int x=0, y=0;  
    int sortie=1;  
  
    clrscr ();  
  
    printf ("Saisie de x:\n");  
    scanf ("%d",&x);  
    printf ("Saisie de y:\n");  
    scanf ("%d",&y);  
  
    do  
    {  
        clrscr ();  
  
        gotoxy (x,y);  
        printf ("#");  
  
        car=getch ();  
        car = toupper (car);    /* Evite le test minuscule/majuscule */  
  
        switch (car)  
        {  
            case 'G':  
                x--;  
                if (x<1)  
                    x = 80;  
                break;  
  
            case 'D':  
                x++;  
                if (x>80)  
                    x = 1;  
                break;  
  
            case 'H':  
                y--;  
                if (y<1)  
                    y = 25;  
                break;  
  
            case 'B':  
                y++;  
                if (y>25)  
                    y = 1;  
                break;  
  
            case 'S':
```

```
        sortie=0;
        break;

    default:
        printf ("%c",0x07);
        break;
    }
} while (sortie==1);
return (0);
}
```

! **La cerise sur le gâteau ...**

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

int main ()
{
    char car;
    int y=10,x=10;
    int sortie = 0;

    clrscr ();
    do
    {
        car=getch ();
        switch (car)
        {
            case 0:
                car=getch();
                switch(car)
                {
                    case 'H':
                        y--;
                        break;
                    case 'P':
                        y++;
                        break;
                    case 'K':
                        x--;
                        break;
                    case 'M':
                        x++;
                        break;
                }
                break;

            case 'S':
            case 's':
                sortie = 1;
                break;
        }

        clrscr ();
        gotoxy (x,y);
        cprintf ("*");

    } while (!sortie);

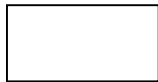
    return (0);
}
```

8 Organigrammes et algorithmes.

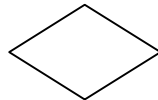
8.1 Organigrammes.

Bien qu'inusité de nos jours, les organigrammes permettent d'avoir une approche visuel que ne fournissent pas les algorithmes. Un organigramme se définit comme une représentation graphique des structures de contrôle.

Nous ne verrons que les deux symboles suivants :



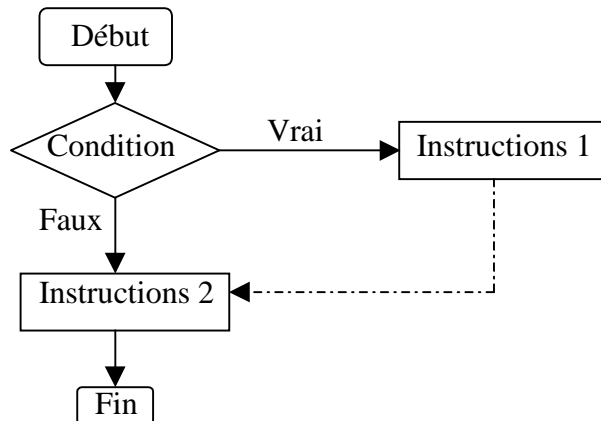
Le rectangle qui représente un bloc d'instructions.



Le losange qui représente une condition : Si Alors Sinon ...

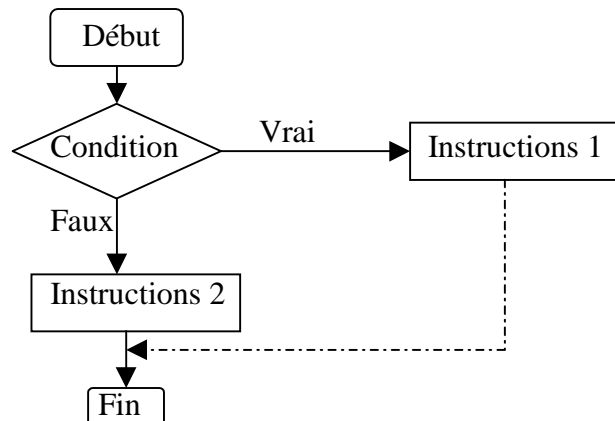
8.1.1 If (Condition) Alors Instructions 1

```
if (Condition)
{
    Instructions 1;
}
Instructions 2;
```

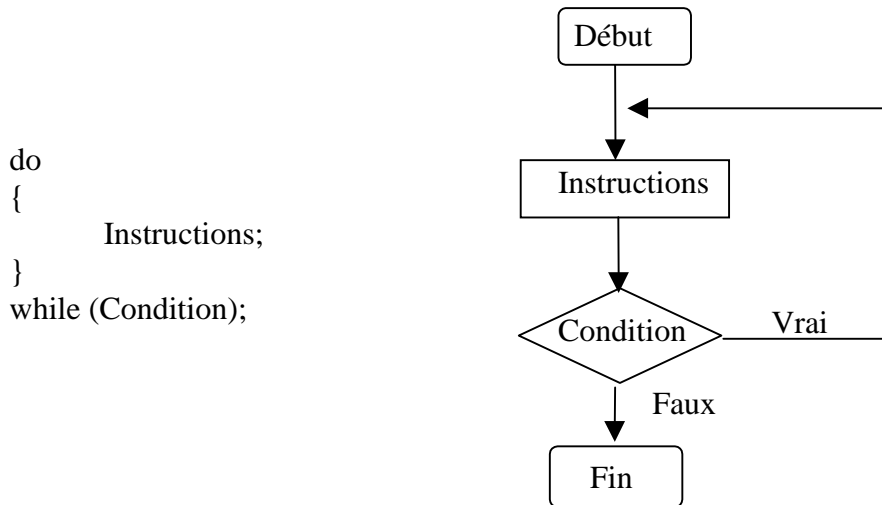


8.1.2 If (Condition) Alors Instructions 1 Else Instructions 2

```
if (Condition)
{
    Instructions 1;
}
else
{
    Instructions 2;
}
```



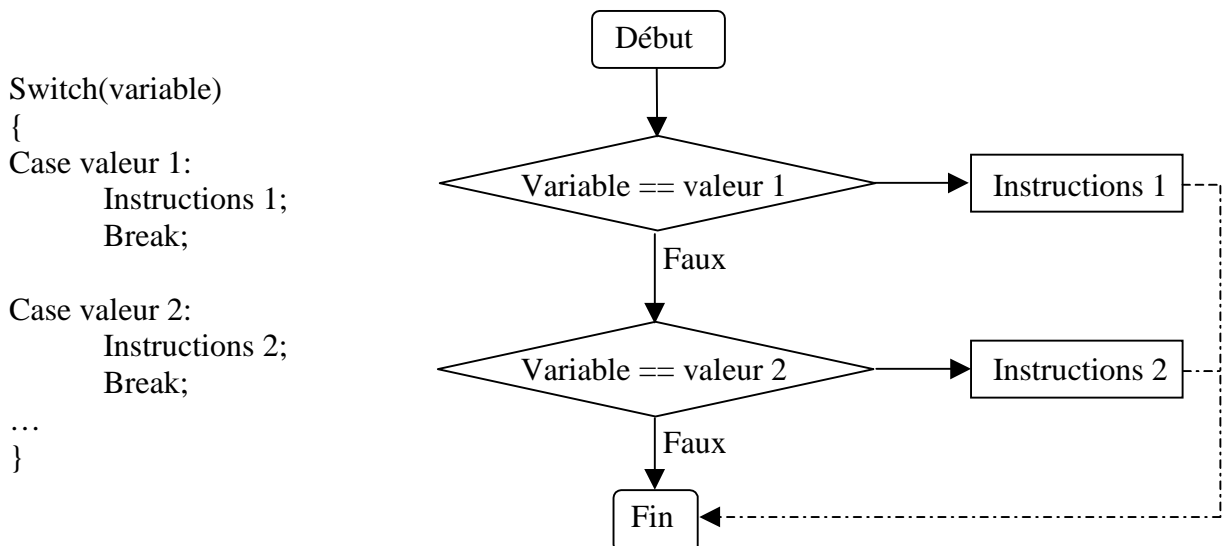
8.1.3 Do ... While (Condition)



8.1.4 While (Condition)

Je ne détaillerai pas cet algorithme au vue qu'il est très peu différent de celui-ci dessus et constituera un excellent exercice.

8.1.5 Switch (variable)... Case ...



8.1.6 Conclusion sur les organigrammes

Les organigrammes permettent de construire des programmes à l'aide de dessins, ceci simplifie énormément la tâche d'analyse du programmeur. Il a été maintenant remplacé par les algorithmes qui en deux mots peuvent être définis comme une transcription francisée de ce que l'on désire faire (pour i allant de 1 à 10 faire ...). Les algorithmes seront vus succinctement dans l'initiation au C niveau 2.

8.2 Exercice 1 : Et les shadoks crièrent : "Pourquoi tant que on a pas tout compris, on continue ?"

Traduire en langage C, compilez, exécutez, comprenez ...

- Déclarez une variable nb_rech de type entier
 - Déclarez une variable nb_user de type entier
 - Déclarez une variable essai de type entier et l'assigner à la valeur 0
 - Déclarez une variable gagne de type entier et l'assigner à la valeur 0
 - Assignez à la variable nb_rech la valeur de random (101)
 - Effacez l'écran
 - Faire
 - {
 - Demandez et saisissez un nombre entre 0 et 100. Soit nb_user la variable utilisée.
 - Si nb_user < nb_rech
 - {
Afficher « Le nombre à trouver est plus grand »
essai = essai + 1
}
 - Sinon
 - {
Si (nb_user > nb_rech)
 - {
Afficher « Le nombre à trouver est plus petit »
essai = essai + 1
}
 - Sinon
gagne = 1;
- Si (gagne)
Ecrire "Gagné"
Sinon
 - {
Ecrire "Perdu"
Faire afficher le nombre nb_rech
}
- Attendre l'appui d'une touche

8.3 Et les shadoks découvrirent la couleur

La fonction **textcolor** (**no_couleur**) permet de changer la couleur du texte. Le no de couleur (**no_couleur**) est un entier allant de 0 à 15. Pour pouvoir afficher en couleur, il est nécessaire d'utiliser la fonction **cprintf** à la place de **printf**. Cprintf possède la même syntaxe que **printf**.

<i>Constante</i>	<i>Couleur</i>	<i>Valeur</i>	<i>Couleur de Fond</i>	<i>Couleur de caractère</i>
BLACK	Noir	0	Oui	Oui
BLUE	Bleu	1	Oui	Oui
GREEN	Vert	2	Oui	Oui
CYAN	Cyan	3	Oui	Oui
RED	Rouge	4	Oui	Oui
MAGENTA	Magenta	5	Oui	Oui
BROWN	Brun	6	Oui	Oui
LIGHTGRAY	Gris Clair	7	Oui	Oui
DARKGRAY	Gris Foncé	8	Non	Oui
LIGHTBLUE	Bleu Clair	9	Non	Oui
LIGHTGREEN	Vert Clair	10	Non	Oui
LIGHTCYAN	Cyan Clair	11	Non	Oui
LIGHTRED	Rouge Clair	12	Non	Oui
LIGHTMAGENTA	Magenta Clair	13	Non	Oui
YELLOW	Jaune	14	Non	Oui
WHITE	Blanc	15	Non	Oui
BLINK	Clignotant	128	Non	***

8.4 Utilisation du clignotement et des codes couleur.

Les valeurs du tableau précédent s'utilisent soit par leur nom, soit par leur code couleur. Ceci signifie que :

textcolor (2) est équivalent à textcolor (GREEN);

En fait GREEN est défini comme valant 2. (cf. #define dans Initiation au C niveau 2).

Pour faire clignoter du texte, il faut ajouter la valeur BLINK à la valeur de la couleur ce qui nous donne :

```

ou          textcolor (2+128);
ou          textcolor (GREEN+BLINK);
ou encore   textcolor (130);
```

Exemple :

```
#include <conio.h>

int main ()
{
    clrscr ();
    textbackground (GREEN);
    textcolor (RED);
    printf ("Coucou");
    getch ();
}
```

affiche "Coucou" en couleur rouge sur fond vert.

Test :

```
Au lieu de
    clrscr ();
```



```
textbackground (GREEN);  
  
écrivez :  
textbackground (GREEN);  
clrscr ();  
Aucun commentaire ...
```

8.5 Exercice 2 :

Ecrivez un programme qui fait afficher "Je suis vert de peur" en bleu clignotant sur fond jaune poussin. Eh oui, pas shadok pour rien.

8.6 Exercice 3 : Et les shadoks créèrent un mini – yam ...

Je dispose de 5 dés numérotés de 1 à 5.

Soient d1, d2, d3, d4, d5 ces 5 dés.

Utilisez la fonction Random pour afficher le tirage des 5 dés.

Affichez gagné si le résultat est un Yam (5 dés identiques).

Corrigés des exercices du chapitre 8

! **Exercice 1 : Et les shadoks crièrent : "Pourquoi tant que on a pas tout compris, on continue ?"**

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

int main ()
{
    int nb_rech=0, nb_user=0;
    int gagne=0;
    int essai=0;

    randomize ();
    nb_rech= random (101); /* Nombre entre 0 et 100 */

    clrscr ();

    do
    {
        printf("Saisir un nombre entre 0 et 100 :");
        scanf ("%d",&nb_user);

        if (nb_user<nb_rech)
        {
            printf("Le nombre à trouver est plus grand\n");
            essai=(essai+1);
        }
        else
        {
            if(nb_user>nb_rech)
            {
                printf("Le nombre à trouver est plus petit\n");
                essai=(essai+1);
            }
            else
            {
                gagne=1;
            }
        }
    }while((essai!=5)&&(!gagne));

    if (gagne)
        printf("Gagné !");
    else
        printf("Perdu !");

    printf("\nNombre recherché : %d",nb_rech);

    getch ();
}
```

! **Exercice 2**

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
```

```
int main()
{
    clrscr ();
    gotoxy(20,20);
    textbackground (YELLOW);
    textcolor (BLUE + BLINK);
    cprintf ("Je suis vert de peur");
    getch ();
}
```

! Exercice 3 : : Et les shadoks créèrent un mini – yam ...

```
#include <stdio.h>
#include<stdlib.h>

int main ()
{
    int d1,d2,d3,d4,d5;

    randomize ();

    /* Tirage des 5 dés */
    d1 = random (6) + 1;
    d2 = random (6) + 1;
    d3 = random (6) + 1;
    d4 = random (6) + 1;
    d5 = random (6) + 1;

    clrscr ();

    /* Affichage du tirage */
    printf ("\nD1 : %d, D2 : %d, D3 : %d, D4 : %d, D5 : %d"
    ,d1,d2,d3,d4,d5);

    /* Résultat */
    if ((d1 == d2) && (d2 == d3) && (d3 == d4) && (d4 == d5))
        printf ("\nGagné !");
    else
        printf ("\nPerdu !");

    getch ();
}
```

9 Les boucles

9.1 Et les shadoks pédalèrent pendant 15 tours ...

Pour faire effectuer un certain nombre de fois une tâche on utilise l'instruction `for` de la façon suivante (avec `i`, une variable de type entier (`int`)).

```
for (i=point de départ; i<point d'arrivée; i=i+pas)
{
    instruction(s) répétée(s);
}
```

Pour un souci de simplicité, nous dirons simplement que la formule suivante :

```
for (i=0; i<15; i++)
{
    instr;
}
```

signifie que l'on va exécuter *instr* pour `i` valant 0 à 14 (<15) c'est à dire 15 fois.

Exemple:

```
#include <conio.h>

int main ()
{
    int i;

    clrscr ();

    for (i=0; i<15; i++)
    {
        printf ("Je me répète pour i valant %d\n",i);
    }

    printf ("Je me suis répété, 15 fois");
    getch ();
}
```

9.2 Syntaxe

De la même façon que le **if** le **for** ne nécessite pas d'accolade si le nombre d'instructions à répéter est de 1.

Exemple :

On peut utiliser cette fonctionnalité dans le programme précédent en remplaçant :

```
for (i=0; i<15; i++)
{
    printf ("Je me répète pour i valant %d\n",i);
}
```

par

```
for (i=0; i<15; i++)
    printf ("Je me répète pour i valant %d\n",i);
```

9.3 Exercice 1

Utilisez une boucle pour `i` variant de 0 à 15 inclus pour afficher :

"Ceci est la couleur i"

où i est remplacé par sa valeur en position (1,i+1) avec la couleur i.

●* cprintf ne supporte pas le \n permettant de retourner à la ligne; de ce fait utilisez i allant de 0 à 15 pour modifier le no de ligne.

Aide

```
gotoxy (1,i+1); /* +1 car le coin haut gauche est aux coordonnées (1,1) */
textcolor (i);
cprintf ("Ceci est la couleur %d",i);
```

9.4 Notion de double boucle.

Il est possible de remplacer les instructions par une boucle afin de réaliser une double boucle.

On obtient donc :

Pour i allant de ... à ...

```
{
    ...
    Pour j allant de ... à ...
    {
        ...
    }
}
```

Exemple :

```
#include <conio.h>

int main ()
{
    int i;
    int j;

    clrscr ();

    for (i=0; i<5; i++)
    {
        printf ("\nJe suis dans la boucle i, i vaut %d\n",i);

        for (j=3; j>0; j--)
        {
            printf ("Je suis dans la boucle j, j vaut %d\n",j);
        }
    }

    getch (); /* Attendre l'appui d'une touche */
}
```

9.5 Exercice 2:

En utilisant la double boucle, écrire un programme qui écrit 1 étoile, saute une ligne, écrit 2 étoiles, saute une ligne, écrit 3 étoiles ... jusqu'à 5 étoiles afin d'obtenir ceci :

```
*
**
***
```

```
****  
*****
```

9.6 Exercice 3 : Et les shadoks fêtèrent Noël ...

9.6.1 "Cône" du sapin

A l'aide d'une double boucle et d'un gotoxy, réaliser un cône pour dessiner le haut du sapin en vert sur 12 lignes. Vous devriez obtenir ceci :

```
*  
***  
*****
```

sur 12 lignes ...

Aide

Sur la ligne n° 1, on affiche $1 * 2 - 1 = 1$ étoile en position (42-1,1)

Sur la ligne n° 2, on affiche $2 * 2 - 1 = 3$ étoiles en position (42-2,2)

Sur la ligne n° 3, on affiche $3 * 2 - 1 = 5$ étoiles en position (42-3,3)

Sur la ligne n° i, on affiche $(i * 2 - 1)$ étoile en position (42-i, i);

On obtient donc :

```
for (j=0; j<(i*2-1); j++)  
{  
    gotoxy (42-i+j,i);  
    printf ("*");  
}
```

9.6.2 Affichage du tronc

Pour poursuivre le sapin, il nous faut maintenant dessiner le tronc. Ecrire la suite du programme en dessinant le tronc à l'aide du caractère @. Vous devriez obtenir ceci (en brun) :

```
@ @ @  
@ @ @  
@ @ @
```

9.6.3 Affichage des boules de Noël

Afficher à l'aide de l'option blink des boules de Noël.

9.7 Exercice 4 : Table Ascii

Les codes Ascii (i.e. les nombres qui représentent les caractères en informatique) vont de 0 à 255. Ecrire un programme qui fait afficher à la suite (\n) puis de façon propre c'est à dire dans un tableau de 16 * 16 les codes Ascii avec les caractères qui leur correspondent.

Aide

Pour faire afficher le caractère associé à un code Ascii, on écrit :

```
printf ("%d : %c", code_ascii, code_ascii);
```

Exemple : int i = 65;
 printf ("%d : %c", i, i); affichera 65 : A

Corrigés des exercices du chapitre 9

! **Exercice 1**

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

int main ()
{
    int i;
    clrscr ();

    for (i=0; i<=15; i++)
    {
        textbackground(15-i);
        gotoxy (1,i+1);
        textcolor (128+i);
        cprintf ("Ceci est la couleur %d", i);
    }

    getch ();
}
```

! **Exercice 2**

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

int main ()
{
    int i;
    int j;

    clrscr ();

    for (i=1; i<5; i++)
    {
        for (j=1; j<=i; j++)
        {
            textcolor (11);
            cprintf ("*");
        }
        printf ("\n");
    }

    getch ();
}
```

! **Exercice 3 : Et les shadoks fêtèrent Noël ...**

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

int main ()
{
    int i;
```

```
int j;

clrscr ();

for (i=1; i<=12; i++)
{
    for (j=1; j<= (i*2-1); j++)
    {
        gotoxy (41-i+j,i);
        textcolor (2);
        cprintf ("*");
    }

    printf ("\n");
}

for (i=1; i<=3; i++)
{
    for (j=3; j>=1; j--)
    {
        gotoxy (39+j,12+i);
        textcolor (6);
        cprintf ("@");
    }
}

textcolor (130);
gotoxy (41,3);
cprintf ("*");

getch ();
}
```

! **Exercice 4 : Table Ascii**

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

void main ()
{
    int i;
    int j;

    clrscr ();

    for (i=0; i<255; i++)
    {
        printf ("%d -> %c\n",i,i);
    }

    getch ();
}
```

Attention, certains code Ascii provoquent des problèmes d'affichage ...

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

void main ()
{
    int i;
```



```
int j;
int car = 0;

clrscr ();

printf (" 0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15\n");
for (i=1; i<=16; i++)
{
    for (j=1; j<=16; j++)
    {
        printf ("%c  ",car);
        car ++;
    }
    printf ("\n ");
}

getch ();
}
```

10 Test d'assimilation des connaissances

Vous allez être placé dans les conditions d'un programmeur et non d'un analyste-programmeur. Ceci signifie que presque tous les algorithmes dont vous aurez besoin seront écrits et vous "n'aurez plus qu'à" les traduire. N'hésitez pas à consulter les cours et surtout les corrigés et exemples. Bon courage ...

10.1 Bibliothèques

Nous utiliserons la déclaration des bibliothèques suivantes :

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
```

10.2 But du programme

Le programme que vous allez écrire consiste en une course de 3 chevaux dans laquelle l'utilisateur mise sur un des chevaux.

Aide : Essayez de bien comprendre ce qui est demandé et ce qui est écrit avant de toucher votre clavier

10.3 Variables utilisées

```
/* saisie du pari */
char  car;           /* caractère saisi par l'utilisateur */
int    sortie;      /* booléen pour saisie du pari */
int    pari;        /* cheval sur lequel a parié le joueur */

int    coul1,coul2,coul3; /* couleurs des chevaux */

/* dessin de la piste */
int    i;           /* variable de boucles */
int    x1=0,x2=0,x3=0; /* La position des 3 chevaux */
int    avance = 0; /* l'avance de chacun des chevaux */
time_t t;          /* pour attente (partie offerte) */

int    premier;    /* cheval ayant gagné */
```

10.4 Saisie du no de cheval parié gagnant

Effacer l'écran.

Se positionner en (1,1)

Afficher : Sur quel cheval voulez vous parier (1,2 ou 3) ?

Faire

Sortie = 1 ;

Car = caractère saisi

Dans le cas où car est égal **au caractère** :

```
1 :
    pari = 1;
    coul1 = 2;
    coul2 = 1;
    coul3 = 1;

2 :
    pari = 2;
    coul1 = 1;
    coul2 = 2;
    coul3 = 1;

3 :
    pari = 3;
    coul1 = 1;
    coul2 = 1;
    coul3 = 2;

autrement :
    sortie = 0
    printf ("%c",0x7);
```

Tant Que sortie == 0

10.5 Dessin de la piste

Effacer l'écran

La piste sera dessinée grâce à deux lignes de 80 caractères – placées respectivement en ligne 10 et en ligne 14.

Utilisez une boucle for pour afficher la piste.

10.6 Course des chevaux

```
/* Initialisation des variables aléatoire */
randomize ();
```

```
/* Dessin des chevaux */
```

```
Faire
```

```
{
```

```
    -> Effacement du cheval 1 (position précédente)
    Position en (x1,11)
    Ecrire (" ")
```

```
    -> Affichage du cheval 1
    Couleur de texte : coul1
```

```
    -> Avance du cheval
    x1 = x1 + nombre aléatoire de 1 à 6
```

```
    Positionnement en x1,11
    Affichage du caractère "1"
```

Faire la même chose pour le cheval 2 sur la ligne 12 avec la couleur coul2 et la position de ligne x2.

Aide : le cheval 1 était en ligne 11

Faire la même chose pour le cheval 3 sur la ligne 13 avec la couleur coul3 et la position de ligne x3.

```
/* Partie de code offerte permettant d'attendre un peu */  
for (i=0; i<500; i++)  
    time (&t);  
}
```

tant que aucun cheval n'a franchi la colonne 74.

Aide : le cheval a franchi la colonne 74 quand sa position est supérieure à 74.

10.7 Détermination du gagnant

Effectuez les différents tests pour connaître le cheval arrivé en premier. On affectera la valeur 1 à la variable premier si c'est le 1^{er} cheval qui a gagné (x1 le plus grand), 2 pour le cheval 2 et 3 pour le cheval 3.

Aide : Le cheval arrivé en premier est celui dont la position est la plus grande.

10.8 Pari gagnant ou perdant

Si le cheval parié est le même que le cheval arrivé en premier, afficher :

Bravo vous avez gagné.

Sinon

Désolé vous avez perdu.

Cheval n°... vainqueur

Attendre l'appui d'une touche.

Correction de l'exercice du chapitre 10

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>

int main ()
{
    int    i;
    int    x1=0,x2=0,x3=0;
    int    avance = 0;
    time_t    t;
    int    pari;
    int    premier;
    int    sortie;
    int    coull1,coul2,coul3;
    char    car;

    /* Pari sur un cheval */
    clrscr ();
    gotoxy (1,1);
    printf ("Sur quel cheval voulez vous parier (1,2 ou 3) ?");

    /* Choix d'un cheval */
    do
    {
        sortie = 1;

        car = getch ();

        switch (car)
        {
            case '1':
                pari = 1;
                coull1 = 2;
                coul2 = 1;
                coul3 = 1;
                break;

            case '2':
                pari = 2;
                coull1 = 1;
                coul2 = 2;
                coul3 = 1;
                break;

            case '3':
                pari = 3;
                coull1 = 1;
                coul2 = 1;
                coul3 = 2;
                break;

            default:
                sortie = 0;
                printf ("%c",0x7);
                break;
        }
    }
}
```

```
    }
} while (!sortie);

/* Efface l'écran */
clrscr ();

/* Dessin de la piste */
for (i=1; i<=80; i++)
{
    gotoxy (i,10);
    printf ("-");

    gotoxy (i,14);
    printf ("-");
}

/* Initialisation des variables aléatoire */
randomize ();

/* Dessin des chevaux */
do
{
    /* Effacement du cheval 1 (position précédente) */
    gotoxy (x1,11);
    printf (" ");

    /* Affichage du cheval 1 */
    textcolor (coull);
    x1 += random (6) + 1;
    gotoxy (x1,11);
    cprintf ("1");

    /* Effacement du cheval 2 (position précédente) */
    gotoxy (x2,12);
    printf (" ");

    /* Affichage du cheval 2 */
    textcolor (coul2);
    x2 += random (6) + 1;
    gotoxy (x2,12);
    cprintf ("2");

    /* Effacement du cheval 3 (position précédente) */
    gotoxy (x3,13);
    printf (" ");

    /* Affichage du cheval 3 */
    textcolor (coul3);
    x3 += random (6) + 1;
    gotoxy (x3,13);
    cprintf ("3");

    /* Attente */
    for (i=0; i<5000; i++)
        time (&t);
}
while ((x1<74) && (x2<74) && (x3<74));

if ((x1>x2) && (x1>x3))
    premier = 1;
else
```

```
{
    if ((x2>x1) && (x2>x3))
        premier = 2;
    else
    {
        if ((x3>x1) && (x3>x2))
            premier = 3;
        else
            premier = 0;
    }
}

gotoxy (1,14);
if (premier == pari)
    printf ("\nBravo vous avez gagné.");
else
{
    if (premier == 0)
        printf ("\nDésolé, il y a égalité entre deux chevaux...");
    else
        printf ("\nDésolé vous avez perdu.\nCheval n°%d
vainqueur",premier);
}

getch ();
return (0);
}
```

M. BERTHOMIER ERIC

INITIATION AU LANGAGE C NIVEAU 2

APPRENTISSAGE DE LA PROGRAMMATION

Version 1.1 du 9 Août 2000

1 Pointeurs et Fonctions

1.1 Variables : pointeurs et valeurs

1.1.1 Les variables et la mémoire.

Une variable (par exemple char car) est en fait une petite zone mémoire ici de 1 octet que l'on s'alloue et où l'on va ranger les informations (c'est la boîte vue durant le niveau 1). Chaque type de variable utilise au moins 1 octet. Le type int varie selon les compilateurs, pour nous il utilise 2 octets c'est à dire 2 cases mémoire. Nous avons vu précédemment qu'il était possible de voir le contenu d'une variable avec la fonction printf et qu'il était possible de mettre une valeur dans une variable avec l'opérateur = ou la fonction scanf.

Exemple :

```
char car = 'C';  
printf ("%c",car);
```

Représentons-nous la mémoire comme une rue remplie de maisons. Chaque case mémoire est représentée par une maison.

Chaque maison porte un numéro, c'est ce que l'on appelle son adresse postale. Pour une case mémoire, on parlera d'adresse mémoire.

Cette adresse est unique pour chaque maison. Cependant, rien ne vous empêche de vous tromper intentionnellement ou non de maison. De la même façon, l'adresse mémoire d'une case est unique mais rien ne vous empêche de vous tromper intentionnellement ou non de case mémoire. (cf. Niveau 1)

Une adresse mémoire :

Exemple vie courante :

Mr Leneuf adresse : 99, grand rue

Exemple en C :

char car adresse : 0x001F (soit 31 en décimal)

0x signifie adresse hexadécimale (ou en base 16).

En langage C, l'adresse mémoire est accessible en faisant précéder la variable de l'opérateur **&**.

Exemple :

```
char car = 'C';  
int nbre = 12;  
  
printf ("Adresse de car : %ld [%lx]",&car, &car);  
printf ("Adresse de nbre : %ld [%lx]",&nbre, &nbre);
```

On ajoute "l" devant le d (%ld) et devant le x (%lx) afin de faire afficher un entier long.

1.1.2 Pointeurs

Pour utiliser les adresses mémoire des variables à la place des variables elles-mêmes, on utilise les pointeurs. Les pointeurs sont définis par un type et se déclarent de la façon suivante :

type* variable;
ou
type *variable;

Le signe * indique l'utilisation d'un pointeur i.e. l'utilisation d'une adresse mémoire. Le type permet simplement de savoir comment le C doit interpréter l'adresse mémoire lors de sa lecture. Lors de l'utilisation des chaînes de caractères nous en verrons des exemples.

Pour enregistrer une valeur dans une adresse mémoire on écrit :

***variable = <valeur>**

*variable signifie contenu de l'adresse mémoire.

Exemple:

```
#include <stdio.h>
#include <stdlib.h>

int main ()
{
    char car='C';
    char* ptr_car = NULL;

    clrscr ();
    printf ("Avant, le caractère est : %c",car);
    ptr_car = &car;      /* ptr_car = adresse de car */
    *ptr_car = 'E';      /* on modifie le contenu de l'adresse mémoire */
    printf ("\nAprès le caractère est : %c",car); /* i.e. on a modifié car */

    getch ();
    return (0);
}
```

NULL signifie l'adresse 0. Il indique que l'adresse mémoire est invalide.

Explicitons cet exemple à l'aide de notre rue bordée de maison.

ptr_car = &car; Signifie que l'on m'a donné l'adresse postale de M. car.
*ptr_car = 'E'; Signifie que je rentre chez M. car et que j'y dépose le caractère E.
printf ("%c",car); Signifie que l'on va lire ce qu'il y a dans la maison de M. et qu'on le fait afficher.

1.1.3 Exercices d'application

Réaliser un programme équivalent qui change la valeur en K.

Réaliser un programme équivalent qui change une valeur numérique (int) de 10 à 35.

1.2 Les fonctions

1.2.1 Définition générale



Une fonction est un petit bloc de programme qui à l'image d'une industrie va créer, faire ou modifier quelque chose.

Un bloc de programme est mis sous la forme d'une fonction si celui-ci est utilisé plusieurs fois dans notre code (dans notre programme) ou simplement pour une question de clarté (Imaginez un livre sans paragraphe).

A l'identique de notre fonction principale `main ()`, une fonction s'écrit de la façon suivante :

```
<type de sortie> <nom de la fonction> (<paramètres d'appels>)  
{  
    Déclaration des variables internes à la fonction  
  
    Corps du programme  
  
    Retour  
}
```

1.2.2 Void

Le type **void** signifie indéfini, il est utilisé :

- comme type de sortie pour les fonctions qui ne retournent aucun résultat.
- dans la déclaration de pointeurs sur des zones mémoires dont on ne connaît pas le type.

Exemples

```
/* Pointeur sur une zone mémoire indéfinie */  
void* m_ptr;  
  
/* Fonction bête affichant un caractère */  
void Affiche_car (char car)  
{  
    printf ("%c",car);  
}
```

1.2.3 Variables globales et locales

Les variables déclarées dans les fonctions sont dites **locales**. Il existe aussi les variables dites **globales** qui sont déclarées en dehors de toute fonction y compris le `main ()`.

Les **variables globales** sont modifiables et accessibles par toutes les fonctions sans avoir besoin de les passer en paramètre. Il est de ce fait extrêmement dangereux d'utiliser des variables globales.

Les **variables locales** ne sont modifiables et accessibles que dans la fonction où elles sont déclarées. Pour les modifier ou les utiliser par une autre fonction, il est nécessaire de les passer en paramètres.

Exemple de variable locale

```
int carre (int val)
{
    int val_retour = 0; /* Déclaration variable locale */
    val_retour = val * val;
    return (val_retour);
}
void main ()
{
    int val_retour = 0; /* Déclaration variable locale */

    val_retour = carre (2);
    printf ("Le carré de 2 est : %d", val_retour);
    getch ();
}
```

`val_retour` est dans les deux cas une **variable locale**. Bien qu'elles aient le même nom dans la fonction `main` et `carre`, **les deux variables n'ont rien à voir entre elles**.

Exemple de variable globale

●* Il est fortement déconseillé d'utiliser les variables globales. ●*

```
#include ...

int    val = 0;      /* Déclaration variable globale */

int carre ()
{
    int    val_retour = 0;
    val_retour = val * val;
    return (val_retour);
}

void main ()
{
    val = 2;
    carre ();
    printf ("Le carre de 2 est %d", carre ());
}
```

`val` est une variable globale.

On constate que le programme devient rapidement illisible ...

1.2.4 Utilisation et modification de données dans les fonctions

L'appel d'une fonction peut s'effectuer à l'aide de paramètres.

- Ces paramètres peuvent être utilisés en les déclarant dans les parenthèses

Exemple

```
void Affiche_Coucou (int x, int y)
{
    gotoxy (x,y);
    printf ("coucou");
}

void main ()
{
    Affiche_coucou (10,12);
    getch ();
}
```

- Ces paramètres peuvent être modifier à la condition qu'ils soient passés par adresse c'est à dire que la fonction reçoive un pointeur.

Exemple

```
void Avance_Position (int* x)
{
    *x = *x + random (6);
}

void main ()
{
    int    i=0;
    int    x=0;

    printf ("Position actuelle : %d",x);

    for (i = 0; i<5; i++)
    {
        Avance_Position (&x);
        printf ("Nouvelle position : %d",x);
    }

    getch ();
}
```

1.2.5 Piège !

Attention, les opérateurs unaires (avec une seule opérande) sur les pointeurs sont dangereux.

En effet :

```
int*    x;
*x++;
```

augmentera l'adresse de x (on va chez le voisin) et non la valeur, pour cela il faut écrire :

```
(*x)++;
```

Ceci est une faute courante, prenez garde ...

1.2.6 Aide pour l'éditeur Turbo C 2.0

Le Turbo C permet de réaliser des copier coller dont vous aurez besoin dans les exercices suivants. Pour se faire, il faut sélectionner un bloc (un bloc est un morceau de texte de l'éditeur, il est marqué en surbrillance).

Note : Ctrl Lettre Lettre s'opère en appuyant sur la touche Ctrl et sans lâcher cette touche en appuyant successivement sur les deux lettres.

Pour sélectionner un bloc :

1. Utilisez la combinaison de touches Ctrl K B, pour marquer le début du bloc.
2. Utilisez la combinaison de touches Ctrl K K pour marquer la fin du bloc.
3. Votre zone est maintenant en surbrillance.

Pour désélectionner un bloc :

Utilisez la combinaison de touches Ctrl K K au début du bloc en surbrillance.

Positionnez vous à l'endroit où vous désirez faire la copie et faites Ctrl K C

Pour déplacer faites Ctrl K V

1.2.7 Exercices

Réalisez une fonction `Avance_cheval` qui :

- ≡ prendra pour paramètres la position x, la position y, la couleur et le caractère représentant le cheval.
- ≡ avancera le cheval en l'effaçant, le dessinant et mettant à jour sa position.

Augmenter le nombre de chevaux à 5.

Correction de l'exercice du chapitre 1

En **gras** apparaissent les principaux changements opérés depuis le programme du niveau 1 chapitre 10.

! **Exercices 1.1.3**

```
#include <stdio.h>
#include <stdlib.h>

int main ()
{
    char car='C';
    char* ptr_car = NULL;

    clrscr ();
    printf ("Avant, le caractère est : %c",car);
    ptr_car = &car;
    *ptr_car = 'K';
    printf ("\nAprès le caractère est : %c",car);

    getch ();
    return (0);
}
```

```
#include <stdio.h>
#include <stdlib.h>

int main ()
{
    int val = 10;
    int *ptr_val = NULL;

    clrscr ();
    printf ("Avant, la valeur est : %d",val);
    ptr_val = &val;
    *ptr_val = 35;
    printf ("\nAprès la valeur est : %d",val);

    getch ();
    return (0);
}
```

! **Fonction Avance_cheval**

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>

void Avance_cheval (int *pos_x, int pos_y, int coul, char cheval)
{
    /* Effacement du cheval (position précédente) */
    gotoxy (*pos_x,pos_y);
    printf (" ");

    /* Affichage du cheval */
    textcolor (coul);
```

```
    *pos_x += random (6) + 1;
    gotoxy (*pos_x,pos_y);
    cprintf ("%c",cheval);
}

int main ()
{
    int i;
    int x1=0,x2=0,x3=0;
    time_t t;
    int pari;
    int premier;
    int sortie;
    int coull,coul2,coul3;
    char car;

    /* Pari sur un cheval */
    clrscr ();
    gotoxy (1,1);
    printf ("Sur quel cheval voulez vous parier (1,2 ou 3) ?");

    /* Choix d'un cheval */
    do
    {
        sortie = 1;

        car = getch ();

        switch (car)
        {
            case '1':
                pari = 1;
                coull = 2;
                coul2 = 1;
                coul3 = 1;
                break;

            case '2':
                pari = 2;
                coull = 1;
                coul2 = 2;
                coul3 = 1;
                break;

            case '3':
                pari = 3;
                coull = 1;
                coul2 = 1;
                coul3 = 2;
                break;

            default:
                sortie = 0;
                printf ("%c",0x7);
                break;
        }
    } while (!sortie);

    /* Efface l',cran */
    clrscr ();
}
```



```
/* Dessin de la piste */
for (i=1; i<=80; i++)
{
    gotoxy (i,10);
    printf ("-");

    gotoxy (i,14);
    printf ("-");
}

/* Initialisation des variables aléatoire */
randomize ();

/* Dessin des chevaux */
do
{
    Avance_cheval (&x1, 11, coul1, '1');
    Avance_cheval (&x2, 12, coul2, '2');
    Avance_cheval (&x3, 13, coul3, '3');

    /* Attente */
    for (i=0; i<5000; i++)
        time (&t);
}
while ((x1<74) && (x2<74) && (x3<74));

if ((x1>x2) && (x1>x3))
    premier = 1;
else
{
    if ((x2>x1) && (x2>x3))
        premier = 2;
    else
    {
        if ((x3>x1) && (x3>x2))
            premier = 3;
        else
            premier = 0 ;
    }
}

gotoxy (1,14);
if (premier == pari)
    printf ("\nBravo vous avez gagné.");
else
{
    if (premier == 0)
        printf ("\nDésolé il y a égalité");
    else
        printf ("\nDésolé vous avez perdu.\nCheval n°%d vainqueur",premier);
}

getch ();
return (0);
}
```

Nous pouvons remarquer que le programme devient plus propre, plus clair.

! **5 chevaux**

```
#include <stdio.h>
```

```
#include <conio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>

void Avance_cheval (int *pos_x, int pos_y, int coul, char cheval)
{
    /* Effacement du cheval (position pr,c,dente) */
    gotoxy (*pos_x,pos_y);
    printf (" ");

    /* Affichage du cheval */
    textcolor (coul);
    *pos_x += random (6) + 1;
    gotoxy (*pos_x,pos_y);
    cprintf ("%c",cheval);
}

int main ()
{
    int i;
    int x1=0,x2=0,x3=0,x4=0,x5=0;
    time_t t;
    int pari;
    int premier;
    int sortie;
    int coull1,coull2,coull3,coull4,coull5;
    char car;

    /* Pari sur un cheval */
    clrscr ();
    gotoxy (1,1);
    printf ("Sur quel cheval voulez vous parier (1,2,3,4 ou 5) ?");

    /* Choix d'un cheval */
    do
    {
        sortie = 1;

        car = getch ();

        switch (car)
        {
            case '1':
                pari = 1;
                coull1 = 2;
                coull2 = 1;
                coull3 = 1;
                coull4 = 1;
                coull5 = 1;
                break;

            case '2':
                pari = 2;
                coull1 = 1;
                coull2 = 2;
                coull3 = 1;
                coull4 = 1;
                coull5 = 1;
                break;
        }
    }
}
```

```
case '3':
    pari = 3;
    coul1 = 1;
    coul2 = 1;
    coul3 = 2;
    coul4 = 1;
    coul5 = 1;
    break;

case '4':
    pari = 4;
    coul1 = 1;
    coul2 = 1;
    coul3 = 1;
    coul4 = 2;
    coul5 = 1;
    break;

case '5':
    pari = 5;
    coul1 = 1;
    coul2 = 1;
    coul3 = 1;
    coul4 = 1;
    coul5 = 2;
    break;

default:
    sortie = 0;
    printf ("%c",0x7);
    break;
}
} while (!sortie);

/* Efface l',cran */
clrscr ();

/* Dessin de la piste */
for (i=1; i<=80; i++)
{
    gotoxy (i,10);
    printf ("-");

    gotoxy (i,16);
    printf ("-");
}

/* Initialisation des variables aléatoire */
randomize ();

/* Dessin des chevaux */
do
{
    Avance_cheval (&x1, 11, coul1, '1');
    Avance_cheval (&x2, 12, coul2, '2');
    Avance_cheval (&x3, 13, coul3, '3');
    Avance_cheval (&x4, 14, coul4, '4');
    Avance_cheval (&x5, 15, coul5, '5');

    /* Attente */
    for (i=0; i<5000; i++)
```

```
        time (&t);
    }
    while ((x1<74) && (x2<74) && (x3<74) && (x4<74) && (x5<74));

    if ((x1>x2) && (x1>x3) && (x1>x4) && (x1>x5))
        premier = 1;
    else
    {
        if ((x2>x1) && (x2>x3) && (x2>x4) && (x2>x5))
            premier = 2;
        else
        {
            if ((x3>x1) && (x3>x2) && (x3>x4) && (x3>x5))
                premier = 3;
            else
            {
                if ((x4>x1) && (x4>x2) && (x4>x3) && (x4>x5))
                    premier = 4;
                else
                {
                    if ((x5>x1) && (x5>x2) && (x5>x3) && (x5>x4))
                        premier = 5;
                    else
                        premier = 0;
                }
            }
        }
    }

    gotoxy (1,17);
    if (premier == pari)
        printf ("\nBravo vous avez gagné.");
    else
    {
        if (premier == 0)
            printf ("\nDésolé il y a égalité entre deux chevaux.");
        else
            printf ("\nDésolé vous avez perdu.\nCheval n°%d
vainqueur",premier);
    }

    getch ();
    return (0);
}
```

Les test opérés à la fin du programme ne sont pas très réjouissants mais nous sommes là pour apprendre le C et non les méthodes d'optimisation.

2 Tableaux & chaînes de caractères

2.1 Tableaux

2.1.1 Définition

Un tableau est un ensemble d'éléments consécutifs. Celui-ci peut être constitué de plusieurs lignes et colonnes. Nous n'utiliserons dans un premier temps que les tableaux à une seule ligne.

Exemple de tableau de caractères

A	B	C	D	E	F	G	H
case 0	case 1	case 2	case 3	case 4	case 5	case 6	case 7

Les cases d'un tableau sont numérotées à partir de 0.

2.1.2 Déclaration

☺ Un tableau se déclare de la manière suivante :
<type> <nom du tableau> [<taille du tableau>];

Exemples

Déclaration d'un tableau de 10 caractères.
char tab_char [10];

Déclaration d'un tableau de 10 nombres.
int tab_int [10];

☺ Un tableau à plusieurs lignes se déclare de la façon suivante :
<type> <nom du tableau> [<taille 1^{ère} dimension 1>][<taille 2^{nde} dimension>] ...;

Exemple

int table [5] [5]; représente un tableau d'entiers de 5 lignes * 5 colonnes.

2.1.3 Utilisation

On accède à un tableau en l'appelant par son nom et son numéro de case.

Exemple

Nom : tab_char Case : 3
tab_char [3] = 'C';

Nom : tab_int Case : 6
tab_int [6] = 10;

Attention : ⚠* Aucune limite n'est fixée sur un tableau ⚠*,
il vous est donc possible d'écrire à l'extérieur de votre tableau donc chez le voisin ...
c'est l'un des bugs le plus courant de l'informatique

2.1.4 Cas spécifique des chaînes de caractères

Les chaînes de caractères sont des tableaux de caractères suivis d'un 0 binaire (ne pas confondre avec le caractère 0, nous parlons ici du code ascii) qui est considéré lui aussi comme un caractère. Une chaîne s'écrit donc : chaîne + 0.

Exemple

Eric s'écrit dans un tableau de 5 caractères de la façon suivante E r i c 0.

E	r	i	c	0
0	1	2	3	4

2.1.5 Déclaration d'une chaîne de caractères

Une chaîne de caractères se déclare sous la forme d'un tableau de caractères de longueur fixe. Attention, comme signalé auparavant, si vous dépassez la longueur de tableau, vous écrivez chez le copain ...

Exemple

```
char m_chaine [20];
```

permettra d'enregistrer des chaînes de 19 caractères maximum (20-1 pour le 0 de fin de chaîne).

D'autre part, il est possible de déclarer une chaîne de caractères sans en spécifier la longueur de départ de la façon suivante :

```
char chaine [] = "Eric";
```

C'est pratique dans la condition où l'on ne doit pas la réutiliser (on ne connaît sa taille que par son contenu).

2.1.6 Ecriture dans une chaîne de caractères

La première méthode permet de déclarer une chaîne mais non de l'initialiser.

Pour se faire, une méthode consiste à utiliser la fonction `sprintf` de la façon suivante :

```
sprintf (<variable de type chaîne de caractères>, "%s", "<valeur d'init.>");
```

Exemple

```
sprintf (m_chaine, "%s", "Eric");
```

remplira m_chaine par Eric0.

(Le 0 n'est là que pour vous rappeler que l'on a besoin du 0 pour terminer la chaîne)

2.1.7 Affichage d'une chaîne de caractères

Une chaîne de caractères s'affiche grâce à la commande `printf` et le mot clé `%s`.

Exemple

```
printf ("%s", chaine);
```

2.1.8 Longueur d'un chaîne

La longueur d'une chaîne de caractères s'obtient par la fonction `strlen`. Le 0 de fin de chaîne n'est pas compté dans cette longueur.

Exemple

```
char ch [] = "toto" ;
```

```
...
```

```
printf ("La longueur de %s est : %d", ch, strlen (ch)) ;
```

Affichera 4 à l'écran.

2.1.9 Exercices

En utilisant une boucle (for),

≡ Remplissez un tableau de 10 caractères avec les lettres de l'alphabet en commençant par A (code ascii 65).

≡ Faîtes afficher la chaîne de caractères ainsi obtenue (n'oubliez pas de rajouter le 0).

≡ Faîtes afficher chaque caractère du tableau sous la forme "Caractère n° 0 : A".

Rappel :

Je peux écrire `Tab_car [i] = code_ascii;` où `code_ascii` est un entier représentant le code Ascii du caractère désigné.

Aide :

Pour faire afficher 1 caractère, on utilisera une syntaxe du style

```
int pos /* Position dans le tableau */  
printf ("Caractère n° %d : %c",pos, Tab_car [pos]);
```

Un élément de tableau peut être assigné à la valeur 0 de la façon suivant :

```
Tab_car [la bonne position] = 0; .
```

2.2 Gets : Saisie d'une chaîne de caractères

2.2.1 Description

La fonction **gets** permet de saisir une chaîne de caractère validée par un Retour Chariot.

Attention, bien que cette chaîne nécessite d'être validée par un retour chariot, celui-ci n'est pas enregistré dans le tableau de caractères.

2.2.2 Exemple d'utilisation

```
#include <stdio.h>  
void main( void )  
{  
    char line[81];  
    /* 81 : taille arbitraire supposée suffisante */  
    /* Une ligne écran = 80 caractères + 1 case pour le 0 de fin de chaîne */  
  
    printf( "Saisissez une chaîne de caractère :\n" );  
    gets( line );  
    /* La frappe de l'utilisateur sera enregistrée dans line, on suppose qu'il  
    ne frappera pas plus de 80 caractères sinon aïe aïe aïe */  
  
    printf( "\nLa chaîne de caractères saisie est : \n%s\n", line );  
    printf( "Notons qu'il n'y a qu'un retour chariot." );  
}
```

Exemple d'exécution

```
Saisissez une chaîne de caractère :  
Bonjour !
```

La chaîne de caractère saisie était :
Bonjour !

Notons qu'il n'y a qu'un retour chariot.

2.2.3 Passage d'une chaîne de caractères en paramètres

Il est bien agréable de pouvoir utiliser cette fonction mais son efficacité est très limitée. Nous allons donc créer notre propre fonction de saisie. Pour cela, il est nécessaire de passer en paramètre une chaîne de caractères.

Ceci s'effectue de la façon suivante :

```
int ma_saisie (char* chaine)
{
    Faire ce qu'il faut ...

    return (0);
}

void main ()
{
    char ma_chaine [30];

    ma_saisie (ma_chaine);

    return ();
}
```

On ne passe pas ici par le caractère **&** car en fait un tableau est une adresse mémoire sur une suite de cases. Par contre, on peut passer par **&** en faisant `ma_saisie (&chaine [0])`, `&chaine [0]` représentant l'adresse mémoire de la première case mémoire du tableau C'est un peu compliqué mais il suffit de se dire, je passe un tableau donc un ensemble de cases mémoire, donc je ne mets pas de **&**. Je transmets une case donc un élément d'un type comme un autre donc je mets le **&**.

2.3 Bug en chaînes ...

.Reprenez l'exercice 2.1.7 et mettez 12 caractères dans votre tableau de 10 cases.
.Essayez avec 80 caractères.

On peut facilement dire que le cas de dépassement de la taille d'un tableau est le bug le plus fréquent rencontré en programmation C alors prenez gare ...

N.B. Ce problème n'existe pas sous Turbo Pascal mais on ne peut pas concilier sécurité et rapidité ce qui fait que le C est de loin plus rapide que Turbo Pascal. Il suffit de s'imaginer qu'il est nécessaire d'effectuer un test à chaque assignation d'un élément du tableau !!!

2.4 Exercices

Réaliser votre propre fonction de saisie de chaîne de caractères ...

≡ Paramètres d'entrée : position x, y, chaîne de caractères, nombre de caractères maximum.

Algorithme :

```
/* Amorçage de la boucle */  
car = saisie d'un caractère  
tant que  
    (car est différent d'un retour chariot (0x0D)) ET  
    (nombre de caractères saisis < nombre de caractères maximum))  
faire  
    ajouter car dans le tableau  
    afficher car à l'écran  
    mettre à jour les positions dans le tableau et sur l'écran  
    car = saisie_car  
fin du tant que  
ajouter le 0 binaire de fin de chaîne.
```

Ajouter à votre fonction de saisie de caractères la gestion des touches backspace (code Ascii 8) et flèche gauche (code Ascii 0 suivi du code Ascii 75). Ces touches supprimeront le caractère qui précède en l'effaçant.

Pour vous aider, reportez-vous au chapitre 4 du cours Niveau 1.

Correction des exercices du chapitre 2

⌈ **Boucle for**

En utilisant une boucle (for),

Remplissez un tableau de 10 caractères avec les lettres de l'alphabet en commençant par A (code ascii 65).

Faites afficher la chaîne de caractères ainsi obtenue (n'oubliez pas de rajouter le 0).

Faites afficher chaque caractère du tableau sous la forme "Caractère n° 0 : A".

```
#include <stdio.h>
#include <stdlib.h>

void main ()
{
    char tableau [11];      /* 10 caractères + 0 de fin de chaîne */
    int i=0; /* compteur */

    clrscr ();

    /* Remplissage du tableau avec les caractères */
    for (i=0; i<10; i++)
        tableau [i] = 65 + i;

    /* Ajout du 0 de fin de chaine */
    tableau [10] = 0;

    /* Affichage de la chaîne */
    printf ("Tableau : %s\n",tableau);

    /* Saut d'une autre ligne */
    printf ("\n");

    /* Affichage de chacun des caractères */
    for (i=0; i<10; i++)
    {
        printf ("Caractère n°%d : %c\n",i,tableau [i]);
    }

    /* Attente de la saisie d'une touche */
    getch ();
}
```

⌈ **Fonction de saisie (1^{ère} partie)**

Réaliser votre propre fonction de saisie de chaîne de caractère ...

Paramètres d'entrée : position x, y, chaîne de caractères, nombre de caractères maximum.

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

void saisie (int x, int y, char *chaine, int max_car)
{
```

```
char car = ' ';
int nb_car = 0;

gotoxy (x,y);
car = getch ();

while ((car != 0x0D) && (nb_car < max_car))
{
    /* Ajout du caractère dans le tableau */
    chaine [nb_car] = car;

    /* Affichage ... l'écran */
    printf ("%c",chaine [nb_car]);

    /* Mise ... jour de la position dans le tableau */
    nb_car ++;

    /* Mise à jour de la position ... l'écran */
    x++;
    gotoxy (x,y);

    /* Saisie d'un nouveau caractère */
    car = getch ();
}
chaine [nb_car] = 0;
}

void main ()
{
    char chaine [11];
    char message [] = "Votre saisie : ";

    clrscr ();

    gotoxy (1,3);
    printf ("%s", message);

    /* 10 : Laisser de la place pour le 0 binaire de fin de chaîne */
    saisie (strlen (message),3,chaine,10);

    /* Affichage du résultat */
    gotoxy (1,5);
    textcolor (GREEN);
    cprintf ("Votre saisie : %s",chaine);

    /* Attente pour visualisation */
    getch ();
}
```

(Fonction de saisie (2^{nde} partie)

Ajouter à votre fonction de saisie de caractères la gestion des touches backspace (code Ascii) et flèche gauche (code Ascii). Ces touches supprimeront le caractère qui précède en l'effaçant.

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

void beep ()
{
```

```
    sound (1000);
    delay (100);
    nosound ();
}

void saisie (int x, int y, char *chaine, int max_car)
{
    char car = ' ';
    int nb_car = 0;

    gotoxy (x,y);
    car = getch ();

    while ((car != 0x0D) && (nb_car < max_car))
    {
        switch (car)
        {
            case 0:
            {
                car = getch ();

                /* nb_car = 0 : d, but de la chaine */
                if ((car == 75) && (nb_car > 0))
                {
                    x--;
                    nb_car--;
                    gotoxy (x,y);
                    printf (" ");
                    /* printf avance le curseur de 1 position */

                    /* Réaffichage du curseur à la bonne position */
                    gotoxy (x,y);
                }
                else
                {
                    beep ();
                }
            }
            break;

            case 8:
            {
                if (nb_car > 0)
                {
                    x--;
                    nb_car--;
                    gotoxy (x,y);

                    printf (" ");
                    /* printf avance le curseur de 1 position */

                    /* Réaffichage du curseur à la bonne position */
                    gotoxy (x,y);
                }
                else
                    beep ();
            }
            break;

            default:
            {
```

*Initiation au langage C.
Niveau 2.*

```
        /* Ajout du caractère dans le tableau */
        chaine [nb_car] = car;

        /* Affichage à l'écran */
        printf ("%c",chaine [nb_car]);

        /* Mise à jour de la position dans le tableau */
        nb_car ++;

        /* Mise à jour de la position à l'écran */
        x++;
        gotoxy (x,y);
    }
    break;

} /* Fin du switch case */

/* Saisie d'un nouveau caractère */
car = getch ();
}

/* Ajout du 0 binaire de fin de chaîne */
chaine [nb_car] = 0;
}

void main ()
{
    char chaine [11];
    char message [] = "Votre saisie : ";

    clrscr ();

    gotoxy (1,3);
    printf ("%s", message);

    /* 10 : Laisser de la place pour le 0 binaire de fin de chaîne */
    saisie (strlen (message)+1,3,chaine,10);

    /* Affichage du résultat */
    gotoxy (1,5);
    textcolor (GREEN);
    cprintf ("Votre saisie : %s",chaine);

    /* Attente pour visualisation */
    getch ();
}
```

3 Le graphisme (1^{ère} partie)

Ras le bol de l'alphanumérique et du 25 * 80 de grand papa... Vive le VGA enfin c'est déjà démodé mais avec notre Turbo C 2.0 c'est tout ce que nous pouvons faire. A noter tout de même que la plupart des jeux d'action Doom Like ont été écrits en MCGA (300*200*256 couleurs !!!).

3.1 Mise en fonction du graphisme & utilisation de l'aide

3.1.1 Utilisation de l'aide

Nous allons apprendre à utiliser l'aide en ligne de Turbo C 2.0.

L'aide en ligne s'affiche lors de l'appui simultané de Ctrl et F1 sur un mot clé.

Exemple :

La fonction `initgraph` provoque l'affichage de :

```
Help
initgraph : initialise le système graphique.
void far initgraph(int far *graphdriver,
                  int far *graphmode,
                  char far *pathtodriver);
Prototype dans graphics.h
Voir aussi  getgraphmode  restorecrtmode
            closegraph   setgraphbufsize
            detectgraph  registerbgidriver
            _graphgetmem graphresult
            getdrivername installuserdriver
```

Ce que fait notre fonction.

Indique la bibliothèque nécessaire pour l'utilisation de cette fonction.

Liens vers d'autres fonctions qui s'utilise conjointement à la nôtre.

Descriptif d'appel de la fonction

Le préfixe `far` signifie une valeur longue du pointeur (peu importe). Pour notre connaissance générale, il faut simplement savoir que c'est comme si les pointeurs normaux ne pouvaient adresser du courrier que sur la France et les pointeurs `far` sur le Monde. Les pointeurs `far` ont disparu avec les compilateurs 32 bits.

Sinon, on retrouve tous les termes précédemment employés ...

La bibliothèque nécessaire nous indique qu'il nous faudra ajouter :
`#include <graphics.h>` à la liste.

Maintenant à nous de jouer.

3.1.2 Les fonctions à utiliser

Voici les fonctions que l'on va utiliser dans un premier temps :

`initgraph (...)`

```
graphresult()  
closegraph ()
```

Exercice :

En utilisant l'aide en ligne, essayez de comprendre l'utilisation des 3 fonctions.

Explications :

Par défaut, les programmes écrits en C fonctionnent en mode Texte. La fonction `initgraph` permet de passer en mode graphique. Ce mode graphique demeure jusqu'à la rencontre de la fonction `closegraph` qui arrêtera le mode graphique pour repasser en mode Texte. Et la fonction `graphresult` alors ? Elle sert uniquement à savoir si le passage du mode texte au mode graphique s'est bien passée.

3.1.3 Ma fonction d'initialisation du graphisme

Mais où est ce que l'on trouve cela ... Bon, je vais vous livrer la potion magique du programmeur ... Ça ne va pas non ! Bon un morceau ...



La première phase consiste à chercher des exemples.

Pour trouver l'utilisation des fonctions on utilise les aides livrées avec le logiciel de programmation ou les livres comme par exemple "Aide mémoire de C" ou Internet.

Dans le cas présent, vous pouvez ouvrir le fichier `Bgidemo.c` qui se trouve dans le répertoire exemple et regarder. C'est bizarre il y a des ressemblances avec ce qui suit.

La seconde phase consiste à essayer de comprendre puis de faire des tests et enfin de se jeter à l'eau en passant au codage ...

Et ça se passe toujours comme ça !

```
#include <graphics.h>  
#include <conio.h>  
#include <stdio.h>  
#include <stdlib.h>  
  
int Initialize(void)  
{  
    int    GraphDriver = DETECT;    /* Driver de carte graphique */  
    int    ErrorCode;              /* Rapport des codes erreurs */  
    int    GraphMode=0;  
  
    initgraph( &GraphDriver, &GraphMode, "c:\\\\tc" );  
  
    ErrorCode = graphresult();  
    /* Lecture du résultat de l'initialisation */  
  
    if( ErrorCode != grOk )  
    {  
        printf("Erreur de système graphique : %s\n",  
              grapherrormsg(ErrorCode) );  
    }  
}
```

```
        return ( 1 );      /* Problème */
    }
}

int main ()
{
    if (Initialize ())      /* Ouverture du mode graphique */
    {
        printf ("Impossible de passer en mode graphique");
        return (1);
    }

    getch ();
    closegraph ();      /* Fermeture du mode graphique */
    return 0;
}
```

Exercice :

A l'aide de l'aide en ligne comprenez le programme (simple ...).

3.1.4 Les fichiers BGI

Les fichiers BGI (*.BGI) sont des pilotes (non pas des pilotes de courses), mais des pilotes de carte vidéo qui permettent d'aller écrire dans la carte vidéo. Dans le turbo C vous trouverez différents pilotes. Par la fonction Windows et la commande rechercher, réaliser la recherche des fichiers *.BGI en partant du répertoire d'installation (c:\tc).

Nom	Dans le dossier
Att.bgi	C:\TC
Cga.bgi	C:\TC
Egavga.bgi	C:\TC
Herc.bgi	C:\TC
Ibm8514.bgi	C:\TC
Pc3270.bgi	C:\TC

Je ne connais pas toutes les reliques d'écran et je ne veux pas connaître toutes les reliques d'écran. Je peux tout de même vous dire que :

- ☞ Egavga.bgi sert à l'EGA et le VGA, c'est celui que nous utiliserons.
- ☞ Herc.BGI est utilisée pour les écrans Hercules (noir et blanc).
- ☞ Cga.bgi pour les écrans CGA.

3.2 Un fichier squelette



Ras le bol de retaper toujours le même corps de programme ...

Nous allons devoir faire beaucoup de petits programmes pour effectuer des tests sur les nouvelles fonctions. Je vous conseille donc de sauvegarder le précédent petit programme sous le nom squelet.c (squelette) puis de le réutiliser pour chaque nouveau petit programme, en prenant soin de faire write as (sauvegarder sous) avec le nouveau nom de programme dès le début pour ne pas écraser squelet.c

3.3 Structure d'un programme

Voilà notre squelette de programme écrit mais en général, sans mode graphique, un programme C s'écrit avec le squelette suivant :

Déclaration des bibliothèques

```
#include <stdio.h>
#include <conio.h>
#include "prog.h"
```

Déclaration des variables globales : elles sont à proscrire impérativement.

```
int pas_bien = 0 ;
```

Déclaration des fonctions

```
int toto ()
{
    printf ("Bonjour, cette fonction ne fait pas grand chose !\n") ;
}
```

Fonction principale

```
int main ()
{
    toto () ;
}
```

3.4 Déclaration des bibliothèques

#include "... .h" indique que nous allons utiliser un fichier .h qui se trouve dans notre répertoire.

#include <... .h> indique que nous allons utiliser un fichier .h qui se trouve dans les bibliothèques du C. Cet endroit est spécifié dans le menu Options Compiler.

Le sujet est vaste et je ne m'attarderai pas dessus.

3.5 Utilisation du graphisme

3.5.1 Ecriture de texte

La fonction à utiliser pour écrire du texte en mode graphique est **outtextxy**.

Par la fonction Ctrl-F1, j'obtiens son prototype :

```
void far outtextxy (int x, int y, char far* textstring);
```

Exercices

1. Tester la fonction.

Exemple

J'ouvre mon squelette et je rajoute ce qui est en gras

```
int main ()
{
    if (Initialize ()) /* Ouverture du mode graphique */
    {
        printf ("Impossible de passer en mode graphique");
        return (1);
    }
}
```

```
    }  
  
    outtextxy (30,30, "Coucou c'est moi !");  
  
    getch ();  
    closegraph ();  
    return (0);  
}
```

Regardez, comprenez ...

2. Ecrivez en 20,50, Oh C mon beau C, que C beau.

3.5.2 Notations

Vous êtes arrivé maintenant à un stade auquel le C ne doit plus vous effrayer, je vous décris donc les fonctions comme elles le sont dans l'aide en ligne. Je vous rappelle que :

☞ **void** signifie indéfini. void à l'entrée d'une fonction signifie donc qu'elle ne renverra rien.

☞ **far** signifie loin. Une antiquité, faites comme si elle n'existait pas.

3.5.3 Couleur de tracé

La définition de la couleur de tracé est obtenue par la fonction :

```
void far setcolor (int color);
```

3.5.4 Lignes, cercles et rectangles

Les fonctions respectives pour dessiner une ligne, un cercle ou un carré sont :

```
void far line (int x1, int y1, int x2, int y2);  
void far circle (int x, int y, int radius);  
void far rectangle (int left, int top, int right, int bottom);
```

Exercice :

Tester ces 3 fonctions dans un même programme.

3.5.5 Effacement de l'écran

Arghhh !!! Notre vieille fonction clrscr () ne marche plus ... Pour effacer l'écran il faut maintenant utiliser la fonction graphique :

```
void far cleardevice (void);
```

3.6 **Exercice complet et instructif ...**

Ecrire un programme qui propose les 6 choix suivants à l'utilisateur :

1. Dessiner une ligne
2. Dessiner un rectangle
3. Dessiner un cercle
4. Choisir la couleur

9. Sortir

qui saisisse le choix de l'utilisateur et fait afficher ce qui est écrit (dans les cas 1,2,3).

Correction des exercices du chapitre 3

⌈ Exercice 3.4.1

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>

int Initialize(void)
{
    int    GraphDriver = DETECT;    /* Driver de carte graphique */
    int    ErrorCode;              /* Rapport des codes erreurs */
    int    GraphMode=0;

    initgraph( &GraphDriver, &GraphMode, "c:\\\\tc" );

    ErrorCode = graphresult();
    /* Lecture du resultat de l'initialisation */

    if( ErrorCode != grOk )
    {
        printf("Erreur de système graphique : %s\n",
            grapherrormsg(ErrorCode) );

        return ( 1 );    /* Problème */
    }
}

int main ()
{
    if (Initialize ())    /* Ouverture du mode graphique */
    {
        printf ("Impossible de passer en mode graphique");
        return (1);
    }

    outtextxy (20,50, "Oh C mon beau C, que C beau");

    getch ();
    closegraph ();    /* Fermeture du mode graphique */

    return 0;
}
```

⌈ Exercice 3.4.3

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>

int Initialize(void)
{
    int    GraphDriver = DETECT;    /* Driver de carte graphique */
    int    ErrorCode;              /* Rapport des codes erreurs */
    int    GraphMode=0;
```

```
initgraph( &GraphDriver, &GraphMode, "c:\\tc" );

ErrorCode = graphresult();
/* Lecture du résultat de l'initialisation */

if( ErrorCode != grOk )
{
    printf("Erreur de système graphique : %s\n",
grapherrormsg(ErrorCode ));

    return ( 1 );    /* Problème */
}
}

int main ()
{
    if (Initialize ())    /* Ouverture du mode graphique */
    {
        printf ("Impossible de passer en mode graphique");
        return (1);
    }

    setcolor (7);
    line (10,10,250,170);

    setcolor (3);
    circle (250,350,100);

    setcolor (4);
    rectangle (100,200,350,400);

    getch ();
    closegraph ();    /* Fermeture du mode graphique */

    return 0;
}
```

Exercice 3.5

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>

int Initialize(void)
{
    int    GraphDriver = DETECT;    /* Driver de carte graphique */
    int    ErrorCode;    /* Rapport des codes erreurs */
    int    GraphMode=0;

    initgraph( &GraphDriver, &GraphMode, "c:\\tc" );

    ErrorCode = graphresult();
    /* Lecture du r,sultat de l'initialisation */

    if( ErrorCode != grOk )
    {
        printf("Erreur de système graphique : %s\n",
grapherrormsg(ErrorCode ));
    }
}
```

```
        return ( 1 );    /* Problème */
    }
}

int main ()
{
    char car = ' ';
    int couleur = 1; /* On évite le 0 car noir sur fond noir !!!! */
    int sortie = 0;

    if (Initialize ())    /* Ouverture du mode graphique */
    {
        printf ("Impossible de passer en mode graphique");
        return (1);
    }

    do
    {
        /* Affichage du menu */
        cleardevice ();
        setcolor (couleur);
        outtextxy (0,0,"1. Dessiner une ligne.");
        outtextxy (0,10,"2. Dessiner un rectangle.");
        outtextxy (0,20,"3. Dessiner un cercle.");
        outtextxy (0,30,"4. Changer la couleur.");
        outtextxy (0,50,"9. Sortir.");

        car = getch ();
        switch (car)
        {
            case '1':
                cleardevice ();
                line (100,100,250,370);
                outtextxy (200,400, "Appuyez sur une touche.");
                getch ();
                break;

            case '2':
                cleardevice ();
                rectangle (100,200,350,400);
                outtextxy (200,400, "Appuyez sur une touche.");
                getch ();
                break;

            case '3':
                cleardevice ();
                circle (150,250,100);
                outtextxy (200,400, "Appuyez sur une touche.");
                getch ();
                break;

            case '4':
                couleur ++;
                if (couleur > 15)
                    couleur = 1;
                break;

            case '9':
                sortie = 1;
                break;
        }
    }
}
```

```
        }  
    }  
    while (!sortie);  
  
    closegraph ();    /* Fermeture du mode graphique */  
    return 0;  
}
```

4 Le graphisme (2^{nde} partie)

4.1 Valeur de retour de la fonction main ()

La fonction main peut être déclarée avec une valeur de retour :

```
int main ()
```

ou sans:

```
void main ()
```

Le choix entre ces deux façons réside dans l'utilisation de votre programme :

. le fait de renvoyer une valeur (un code erreur) permet lors de l'exécution de votre programme par un autre programme de savoir si tout s'est bien passé.

. le fait de ne pas renvoyer de code erreur rend impossible le fait de savoir si le programme s'est bien terminé (sauf de manière visuelle (en l'exécutant)).

On utilisera généralement void main () pour des programmes de test et int main () pour des vrais programmes.

4.2 Motif de remplissage et contour

Dans tous les exemples qui suivront, j'utiliserai la fonction Initialize () écrite dans le précédent cours.

4.2.1 Exemple

Tapez cet exemple et en vous appuyant sur l'aide en ligne, comprenez l'utilisation des fonctions graphiques setlinestyle, setfillstyle, fillellipse.

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<dos.h>
#include<graphics.h>

int main ()
{
    clrscr();

    /* Ouverture du mode graphique */
    if (Initialize ())
        return (1);

    /* Définit la couleur de tracé des traits */
    setcolor (15);

    /* Définit l'épaisseur et le style courants des lignes */
    setlinestyle (DOTTED_LINE, 0, NORM_WIDTH);

    /* Définit le motif et la couleur de remplissage */
    setfillstyle (SOLID_FILL, 12);

    /* Dessine une ellipse de rayon x=15 y=15 donc un cercle rayon 15 */
    fillellipse (30,30,15,15);

    /* Attend l'appui d'une touche : permet de voir ce que l'on fait */
```

```
    getch ();

    /* Termine le mode graphique */
    closegraph();
}
```

4.2.2 Exercices d'application

- 🚧 Changer la couleur du contour
- 🚧 Changer le motif de remplissage.
- 🚧 Changer la couleur du motif de remplissage

4.3 Rectangles et barres

4.3.1 Exemple

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<dos.h>
#include<graphics.h>

int main ()
{
    clrscr();

    /* Ouverture du mode graphique */
    if (Initialize ())
        return (1);

    /* Définit la couleur de tracé des traits */
    setcolor (13);

    /* Définit le motif et la couleur de remplissage */
    setfillstyle (SOLID_FILL, 11);

    /* Barre 3D */
    bar3d (50,50,200,200,13,1);

    /* Attend l'appui d'une touche : permet de voir ce que l'on fait */
    getch ();

    closegraph();
}
```

4.3.2 Explications

void far bar3d (int left, int top, int right, int bottom, int depth, int topflag);

L'aide de **bar3d** n'est pas explicite, en voici donc une explication plus complète (extrait du Manuel de Référence) :

bar3d dessine une barre rectangulaire en trois dimensions (dans l'espace), et la remplit à l'aide du motif et de la couleur de remplissage courants. Le contour dans l'espace de cette barre est tracé dans le style et avec la couleur de tracé courants. La profondeur de la barre est donnée en pixels par l'argument *depth*. L'argument *topflag* sert à

déterminer si la barre possède une face supérieure. Si `topflag` est non nul il y a une face supérieure sinon il n'y en a pas (ce qui permet l'empilage de plusieurs barres). Pour la profondeur de la barre, prendre par exemple 25 % de la barre.

On reconnaît dans `topflag` le fameux type vrai faux (vrai si différent de 0, faux sinon).

4.3.3 Exercices d'application

- ✍️ Changer la couleur de trait
- ✍️ Dessiner un rectangle plein

4.4 Texte graphique : oh les belles lettres ...

4.4.1 Exemple

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<dos.h>
#include<graphics.h>

int main ()
{
    clrscr();

    /* Ouverture du mode graphique */
    if (Initialize ())
        return (1);

    /* Définit la couleur du texte */
    setcolor (10);

    /* Définit le style du texte */
    settextstyle (TRIPLEX_FONT,VERT_DIR,5);

    /* Affiche le texte */
    outtextxy (30,30,"Raven come back");

    /* Attend l'appui d'une touche : permet de voir ce que l'on fait */
    getch ();

    /* Ferme le mode graphique */
    closegraph();
}
```

4.4.2 Taille du texte

Pour connaître la taille des caractères utilisés, nous devons utiliser la fonction :

void gettextsettings (struct textsettingstype far* texttypeinfo) ;

Aïe, quelque chose que je ne connais pas ...

Le mot de syntaxe `struct` nécessiterait un cours à lui tout seul. Nous allons donc utiliser une autre méthode pour connaître la taille des caractères. Relisez l'aide en ligne associée à `settextstyle` et plus spécifiquement l'aide associée au paramètre **charsize**. Eh oui, nous avons

la clé de notre problème de taille de caractères, la taille en pixels des caractères est égal à `charsize * 8`.

Dans l'exemple précédent où `charsize` était de 5 nous avons donc des caractères de 40x40 pixels.

4.4.3 Exercices d'application

- 🐾 Changer le texte ...
- 🐾 Changer la couleur du texte

4.5 Exercice complet et instructif ...




A l'aide de l'exercice 3.4 du cours n°3, réaliser un menu en mode graphique.

Spécifications :

1. Dessiner le menu en mode graphique et en caractères gothiques.
2. Ajoutez un menu texte horizontal qui affiche un texte en horizontal en caractères normaux (DEFAULT_FONT).
3. Ajoutez un menu texte horizontal qui affiche un texte en vertical en caractères normaux (DEFAULT_FONT).

Correction des exercices du chapitre 4

Exercice 4.2.2

-  Changer la couleur du contour
-  Changer le motif de remplissage.
-  Changer la couleur du motif de remplissage

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>

int Initialize(void)
{
    int    GraphDriver = DETECT;    /* Driver de carte graphique */
    int    ErrorCode;              /* Rapport des codes erreurs */
    int    GraphMode=0;

    initgraph( &GraphDriver, &GraphMode, "c:\\\\tc" );

    ErrorCode = graphresult();
    /* Lecture du résultat de l'initialisation */

    if( ErrorCode != grOk )
    {
        printf("Erreur de système graphique : %s\n",
            grapherrormsg(ErrorCode ));
        return ( 1 );    /* Problème */
    }
}

int main ()
{
    char car = ' ';
    int couleur = 1; /* On évite le 0 car noir sur fond noir !!!! */
    int sortie = 0;

    if (Initialize ())    /* Ouverture du mode graphique */
    {
        printf ("Impossible de passer en mode graphique");
        return (1);
    }

    /* Couleur du contour */
    setcolor (8);

    /* Définition du style de ligne */
    setlinestyle (4, 0, NORM_WIDTH);

    /* Définition du type et couleur de remplissage */
    setfillstyle (SLASH_FILL, 2);

    /* Ellipse pleine */
    fillellipse (30,30,15,15);

    /* Attente */
}
```

```
    getch ();  
    closegraph ();    /* Fermeture du mode graphique */  
    return 0;  
}
```


✍ Exercice 4.3.3

- ✍ Changer la couleur de trait
- ✍ Dessiner un rectangle plein

```
#include <graphics.h>  
#include <conio.h>  
#include <stdio.h>  
#include <stdlib.h>  
  
int Initialize(void)  
{  
    int    GraphDriver = DETECT;    /* Driver de carte graphique */  
    int    ErrorCode;    /* Rapport des codes erreurs */  
    int    GraphMode=0;  
  
    initgraph( &GraphDriver, &GraphMode, "c:\\\\tc" );  
  
    ErrorCode = graphresult();  
    /* Lecture du r,sultat de l'initialisation */  
  
    if( ErrorCode != grOk )  
    {  
        printf("Erreur de système graphique : %s\n",  
            grapherrormsg(ErrorCode) );  
  
        return ( 1 );    /* Problème */  
    }  
}  
  
int main ()  
{  
    char car = ' ';  
    int couleur = 1; /* On évite le 0 car noir sur fond noir !!!! */  
    int sortie = 0;  
  
    if (Initialize ())    /* Ouverture du mode graphique */  
    {  
        printf ("Impossible de passer en mode graphique");  
        return (1);  
    }  
  
    /* Couleur du contour */  
    setcolor (4);  
  
    /* D,finition du type et couleur de remplissage */  
    setfillstyle (SLASH_FILL, 4);  
  
    /* Ellipse pleine */  
    bar3d (50,50,200,200,0,1);  
  
    /* Attente */  
    getch ();  
    closegraph ();    /* Fermeture du mode graphique */  
    return 0;  
}
```

Exercice 4.4.3

 Changer le texte ...

 Changer la couleur du texte

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>

int Initialize(void)
{
    int    GraphDriver = DETECT;    /* Driver de carte graphique */
    int    ErrorCode;              /* Rapport des codes erreurs */
    int    GraphMode=0;

    initgraph( &GraphDriver, &GraphMode, "c:\\\\tc" );

    ErrorCode = graphresult();
    /* Lecture du résultat de l'initialisation */

    if( ErrorCode != grOk )
    {
        printf("Erreur de système graphique : %s\n",
            grapherrormsg(ErrorCode) );

        return ( 1 );    /* Problème */
    }
}

int main ()
{
    char car = ' ';
    int couleur = 1; /* On évite le 0 car noir sur fond noir !!!! */
    int sortie = 0;

    if (Initialize ())    /* Ouverture du mode graphique */
    {
        printf ("Impossible de passer en mode graphique");
        return (1);
    }

    /* Couleur du contour */
    setcolor (4);

    /* Définition du type et couleur de remplissage */
    settextstyle (TRIPLEX_FONT, VERT_DIR, 5);

    /* Affichage du texte */
    outtextxy (30,30, "Coucou");

    /* Attente */
    getch ();

    /* Fermeture du mode graphique */
    closegraph ();
    return 0;
}
```

✎ Exercice 4.5

Dessiner le menu en mode graphique et en caractères gothiques.

Ajoutez un menu texte horizontal qui affiche un texte horizontal en caractères normaux.

Ajoutez un menu texte horizontal qui affiche un texte vertical en caractères normaux.

```
#include <graphics.h>
#include <conio.h>
#include <stdio.h>
#include <stdlib.h>

int Initialize(void)
{
    int    GraphDriver = DETECT;    /* Driver de carte graphique */
    int    ErrorCode;              /* Rapport des codes erreurs */
    int    GraphMode=0;

    initgraph( &GraphDriver, &GraphMode, "c:\\\\tc" );

    ErrorCode = graphresult();
    /* Lecture du resultat de l'initialisation */

    if( ErrorCode != grOk )
    {
        printf("Erreur de système graphique : %s\n",
            grapherrormsg(ErrorCode) );

        return ( 1 );    /* Problème */
    }
}

int main ()
{
    char car = ' ';
    int couleur = 1; /* On évite le 0 car noir sur fond noir !!!! */
    int sortie = 0;

    if (Initialize ())    /* Ouverture du mode graphique */
    {
        printf ("Impossible de passer en mode graphique");
        return (1);
    }

    /* Choix de la police de caractère */
    settextstyle (GOTHIC_FONT, HORIZ_DIR, 3);

    /*
    La taille des caractères est maintenant de 3*8 = 24 pixels
    Afin de laisser de la place pour mon rectangle j',cris donc
    une ligne tous les 30 pixels et ... 10 pixels du bord de l',cran
    */

    do
    {
        /* Affichage du menu */
        cleardevice ();
        setcolor (couleur);
        outtextxy (10,30,"1. Dessiner une ligne.");
        outtextxy (10,60,"2. Dessiner un rectangle.");
        outtextxy (10,90,"3. Dessiner un cercle.");
        outtextxy (10,120,"4. Changer la couleur.");
    }
```

*Initiation au langage C.
Niveau 2.*

```
outtextxy (10,150,"5. Texte horizontal.");
outtextxy (10,180,"6. Texte vertical.");
outtextxy (10,250,"9. Sortir.");

car = getch ();
switch (car)
{
    case '1':
        cleardevice ();
        line (100,100,250,370);
        outtextxy (200,400,"Appuyez sur une touche");
        getch ();
        break;

    case '2':
        cleardevice ();
        rectangle (100,200,350,400);
        outtextxy (200,400,"Appuyez sur une touche");
        getch ();
        break;

    case '3':
        cleardevice ();
        circle (150,250,100);
        outtextxy (200,400,"Appuyez sur une touche");
        getch ();
        break;

    case '4':
        couleur ++;
        if (couleur > 15)
            couleur = 1;
        break;

    case '5':
        cleardevice ();

        /* On passe en mode normal */
        settextstyle (DEFAULT_FONT, HORIZ_DIR, 1);
        outtextxy (100,100, "Ceci est un texte");

        /* On revient au style gothique */
        settextstyle (GOTHIC_FONT, HORIZ_DIR, 3);

        outtextxy (200,400, "Appuyez sur une touche");
        getch ();
        break;

    case '6':
        cleardevice ();

        /* On passe en mode normal */
        settextstyle (DEFAULT_FONT, VERT_DIR, 1);
        outtextxy (100,100, "Ceci est un texte");

        /* On revient au style gothique */
        settextstyle (GOTHIC_FONT, HORIZ_DIR, 3);

        outtextxy (200,400, "Appuyez sur une touche");
        getch ();
        break;
}
```

```
                case '9':  
                    sortie = 1;  
                    break;  
            }  
        }  
    while (!sortie);  
  
    closegraph ();    /* Fermeture du mode graphique */  
    return 0;  
}
```


5 Fichiers et Structures

5.1 Bases sur les fichiers

Un fichier représente tout ce qui est enregistré sur votre disque dur ou presque, on va dire tout ce qui porte un nom. Il est possible de créer, de lire ou d'écrire dans des fichiers. Il faut noter cependant que certains fichiers par contre peuvent être protégés en lecture, en écriture ou les deux.

Afin de ne rien endommager dans notre système, créez à l'aide de NotePad ou de la commande Ms-Dos « edit » un fichier nommé test.txt, dans lequel vous taperez un texte de louanges pour ce superbe cours (à votre libre inspiration).

Voici un petit exemple de la lecture du fichier test.txt.

Exemple

```
#include <conio.h>
#include <io.h>
#include <fcntl.h>

int main ()
{
    int h_fic;
    char ligne [80];
    int nb_car_lus;
    int i;

    /* effacement de l'écran */
    clrscr ();

    /* Ouverture du fichier */
    h_fic = open ("c:\\test.txt ", O_CREAT);

    /* Test si fichier ouvert */
    if (h_fic == -1)
    {
        printf ("Impossible d'ouvrir le fichier");
        getch ();
        return (1);
    }

    while (!eof (h_fic))
    {
        /* Lecture de 80 octets maximum */
        nb_car_lus = read (h_fic, ligne, 80);

        /* Ecriture de ce qui a ,t, lu */
        for (i=0; i<nb_car_lus; i++)
        {
            printf ("%c",ligne [i]);
        }
        printf ("\n");
    }

    /* Fermeture du fichier */
    close (h_fic);
}
```

```
/* Ecrire que c'est terminé */
printf ("\n --- Fin ---");

getch ();
return (0);
}
```

Analysons l'ensemble

```
/* Ouverture du fichier */
h_fic = open ("c:\\test.txt", O_CREAT);
```

La fonction **open** permet d'ouvrir un fichier, le second paramètre est son mode d'ouverture, ici O_CREAT signifiant ouverture s'il existe ou création s'il n'existe pas. Par contre O_CREAT seul ne permet pas d'écrire dans le fichier offrant ainsi une sécurité supplémentaire. La fonction open renvoie -1 si l'on a pas réussi à ouvrir notre fichier et renvoie un handle (poignée) sur le fichier sinon.

What is an handle ?

Un lien sur quelque chose. On appelle ceci une poignée car comme avec un sac, vous le prenez par la poignée pour le porter.

Notez la présence de \\ au lieu de \ du fait que \ seul permet la composition de caractère spéciaux (ex \n).

```
/* Test si fichier ouvert */
if (h_fic == -1)
{
    printf ("Impossible d'ouvrir le fichier");
    getch ();
    return (1);
}
```

La fonction **eof** permet de savoir si l'on a atteint la fin du fichier. Regardez l'aide pour comprendre le while (!eof (...))

```
while (!eof (h_fic))
{
```

La fonction **read** permet de lire des octets (ici transformés en char). Elle prend pour paramètre le handle de fichier, la zone de mémoire dans laquelle on va recevoir les données et la taille maximale que l'on veut lire (attention au dépassement (cf. chapitre 2)).

```
/* Lecture de 80 octets maximum */
nb_car_lus = read (h_fic, ligne, 80);

/* Ecriture de ce qui a été lu */
for (i=0; i<nb_car_lus; i++)
{
```

```
        printf ("%c",ligne [i]);
    }
    printf ("\n");
}
```

A ne pas oublier, il faut fermer le fichier après l'avoir ouvert !

```
/* Fermeture du fichier */
```

```
close (h_fic);
```

```
/* Ecrire que c'est terminé */
```

```
printf ("\n --- Fin ---");
```

```
getch ();
```

```
return (0);
```

```
}
```

Très simple non !

5.2 Création d'un autre fichier

Reprenez l'exemple précédent et ajouter ceci après le open :

```
h_fic2= open ("c:\\ma_copie.bat", O_CREAT);
```

N'oubliez pas de déclarer h_fic2.

Ainsi nous allons créer une copie du fichier.

```
int          h_fic2;

/* Création d'une copie */
h_fic2 = open ("c:\\ma_copie.txt", O_CREAT);

/* Test si fichier bien créer */
if (h_fic2 == -1)
{
    printf ("Impossible de créer le nouveau fichier");
    close (h_fic2);
    getch ();
    return (1);
}

while (!eof (h_fic))
{
    /* Coupure */
    printf ("\n");

    /* Ecriture dans notre copie de fichier */
    write (h_fic2, ligne, nb_car_lus);
}

/* Fermeture du fichier */
close (h_fic);

/* Fermeture de ma copie de fichier */
close (h_fic2);
```

En **gras** sont signalés tous les ajouts effectués.

La fonction **write** permet d'écrire des octets sur le fichier. Les paramètres sont le handle de fichiers, les octets à écrire et la taille à écrire.

Essai, laissez 80 au lieu de `nb_car_lus`. Regardez, avec un peu de chance, vous allez trouver des morceaux de texte non désirés, eh oui, personne ne vous a dit que votre chaîne de départ était vide !

5.3 *Ecriture dans un fichier : complément.*

Insérer cette ligne après notre `write` :

```
/* Ecriture dans notre copie de fichier */  
write (h_fic2, ligne, nb_car_lus);
```

oui la même ligne.

Regardez.

La fonction `write` a ajouté à la fin du fichier.

Attention, l'utilisation de `O_CREAT` permet de créer le fichier en mode lecture seul, pour l'ouvrir en plus avec des droits d'écriture il faut lui associer `O_RDWR` (lecture/écriture) ou `O_WRONLY` (écriture seule) de la façon suivante :

```
open ("c:\\test.txt", O_CREAT | O_RDWR).
```

5.4 *Positionnement et taille d'un fichier*

- ☞ La taille d'un fichier est donnée par la fonction **`int filelength (handle)`**
- ☞ La position en octets dans un fichier est donnée par la fonction **`int tell (handle)`**

Exemple

```
#include <conio.h>  
#include <io.h>  
#include <fcntl.h>  
  
int main ()  
{  
    int position;  
    int h_fic;  
    char ligne [80];  
    int nb_car_lus;  
    int i;  
    int taille;  
  
    /* effacement de l'écran */  
    clrscr ();  
  
    /* Ouverture du fichier */  
    h_fic = open ("c:\\test.txt", O_CREAT);  
  
    /* Test si fichier ouvert */  
    if (h_fic == -1)  
    {  
        printf ("Impossible d'ouvrir le fichier");  
    }  
}
```

```
    getch ();
    return (1);
}

/* Taille du fichier */
taille = filelength (h_fic);
printf ("Taille du fichier : %d\n",taille);

while (!eof (h_fic))
{
    /* Position dans le fichier */
    position = tell (h_fic);
    printf ("Position dans le fichier : %d\n",position);

    /* Lecture de 80 octets maximum */
    nb_car_lus = read (h_fic, ligne, 80);

    /* Ecriture de ce qui a ,t, lu */
    for (i=0; i<nb_car_lus; i++)
    {
        printf ("%c",ligne [i]);
    }

    printf ("\n");
}

/* Position dans le fichier */
position = tell (h_fic);
printf ("Position dans le fichier : %d\n",position);

/* Fermeture du fichier */
close (h_fic);

printf ("\n --- Fin ---");

getch ();
return (0);
}
```

5.5 Structures

Nous avons vu, dans le chapitre précédent, l'utilisation des structures. Nous allons maintenant voir la définition et l'utilisation d'une structure.

5.5.1 Déclaration

2 choix se proposent à nous.

Solution 1 :

```
struct
{
    /* Définition de la structure */
} nom de la variable qui aura comme forme cette structure;
```

Solution 2 :

```
typedef struct
{
    /* Définition de la structure */
} nom de la structure;
```

<nom de la structure> nom de la variable qui aura comme structure cette structure.

Je préfère de loin la seconde solution.

Exemple

```
typedef struct
{
    char   nom [40];
    char   prenom [20];
    int    age;
} elt_fiche_nom;

elt_fiche_nom une_fiche ;
elt_fiche_nom une_seconde_fiche ;
```

elt_fiche_nom est le nom de la structure, la macro typedef signifie définition d'un type, de ce fait là elt_fiche_nom se comporte exactement comme un autre type de données (int par exemple).

Par la suite, je n'utiliserai que la seconde méthode.

5.5.2 Utilisation

```
struct
{
    char   nom [40];
    char   prenom [20];
    int    age;
} fiche_nom;

fiche_nom   une_fiche;
fiche_nom   une_seconde_fiche;

une_fiche.age = 20;
```

5.5.3 Taille d'une structure

La taille d'une structure ou d'un type se détermine par la fonction **sizeof**.

Exemple

```
#include <stdio.h>
#include <stdlib.h>
#include <io.h>
#include <fcntl.h>

int main()
{
    typedef struct
    {
        char   nom [40];
        char   prenom [20];
```

```
        int    age;
    } fiche;

    fiche ma_fiche;

    printf ("Taille de la structure : %d\n",sizeof (fiche));
    sprintf (ma_fiche.nom, "%s","BERTHOMIER");
    sprintf (ma_fiche.prenom, "%s","Eric");
    ma_fiche.age = 30;
    printf ("%s %s est âgé de %d",ma_fiche.prenom, ma_fiche.nom,
            ma_fiche.age);

    getch ();
    return (0) ;
}
```

5.6 Fichier & structure

Ce programme ne fonctionne pas sous win98 avec le compilateur Borland C 2. (Compilateur 16 bits)

Il fonctionne par contre avec un compilateur 32 bits (Test avec VC++ 6.0).

L'explication existe mais est hors de portée de ce cours ou alors un bug de ma part. Peut être, peut être. Nul n'est parfait.

Je laisse tout de même celui-ci à titre d'exemple.

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <io.h>
#include <fcntl.h>

int main(int argc, char* argv[])
{
    typedef struct
    {
        char  nom [40];
        char  prenom [20];
        unsigned char  age;
    } fiche;

    fiche ma_fiche;
    int  h_fic ;
    int  i;

    /* Création du du fichier -> le vider s'il existait */
    h_fic = open ("c:\\fiche.dat", O_CREAT | O_TRUNC | O_WRONLY);

    /* Test si fichier ouvert */
    if (h_fic == -1)
    {
        printf ("Impossible d'ouvrir le fichier");
        getch ();
        return (1);
    }

    printf ("Taille de la structure : %d\n\n",sizeof (fiche));

    sprintf (ma_fiche.nom, "%s","BERTHOMIER");
    sprintf (ma_fiche.prenom, "%s","Eric");
    ma_fiche.age = 30;
```

```
printf ("Fiche 1 : %s %s est âgé de %d\n",ma_fiche.prenom,
ma_fiche.nom, ma_fiche.age);
write (h_fic, &ma_fiche, sizeof (fiche));

sprintf (ma_fiche.nom, "%s","P'TIT");
sprintf (ma_fiche.prenom, "%s","Luc");
ma_fiche.age = 48;
printf ("Fiche 2 : %s %s est âgé de %d\n",ma_fiche.prenom,
ma_fiche.nom, ma_fiche.age);
write (h_fic, &ma_fiche, sizeof (fiche));

sprintf (ma_fiche.nom, "%s","Tolkien");
sprintf (ma_fiche.prenom, "%s","JRR");
ma_fiche.age = 89;
printf ("Fiche 3 : %s %s est âgé de %d\n",ma_fiche.prenom,
ma_fiche.nom, ma_fiche.age);
write (h_fic, &ma_fiche, sizeof (fiche));

/* Fermeture du fichier */
close (h_fic);

/* ----- */
/* Lecture des éléments du fichier */
/* ----- */
/* Ouverture du fichier */
h_fic = open ("c:\\fiche.dat", O_RDONLY);

/* Test si fichier bien ouvert */
if (h_fic == -1)
{
    printf ("Impossible d'ouvrir le fichier");
    getch ();
    return (1);
}

for (i=0; i<3; i++)
{
    /* Lecture de la fiche */
    read (h_fic, &ma_fiche, sizeof (fiche));

    /* Affichage des données lues */
    printf ("Fiche n° %d : %s %s est âgé de %d\n", i,
ma_fiche.prenom, ma_fiche.nom, ma_fiche.age);
}

/* Fermeture du fichier */
close (h_fic);

getch ();
return 0;
}
```

5.7 Fin et commencement

Vous disposez maintenant d'à peu près toutes les connaissances nécessaires pour l'élaboration de petits programmes. D'autres cours suivront, en fonction de la demande.

Pour tout commentaire, suggestion de cours, écrivez moi :

eric.berthomier@free.fr ou lesourciergris@free.fr

Annexe 1 : Table Ascii

Ctrl	Dec	Hex	Char	Code	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
^@	0	00		NUL	32	20	sp	64	40	@	96	60	`
^A	1	01	☐	SOH	33	21	!	65	41	A	97	61	a
^B	2	02	☐	SIX	34	22	"	66	42	B	98	62	b
^C	3	03	♥	EIX	35	23	#	67	43	C	99	63	c
^D	4	04	♦	EOI	36	24	\$	68	44	D	100	64	d
^E	5	05	♣	ENQ	37	25	%	69	45	E	101	65	e
^F	6	06	♠	ACK	38	26	&	70	46	F	102	66	f
^G	7	07	+	BEL	39	27	'	71	47	G	103	67	g
^H	8	08	☐	BS	40	28	(72	48	H	104	68	h
^I	9	09	○	HI	41	29)	73	49	I	105	69	i
^J	10	0A	◻	LF	42	2A	*	74	4A	J	106	6A	j
^K	11	0B	♂	VI	43	2B	+	75	4B	K	107	6B	k
^L	12	0C	♀	FF	44	2C	,	76	4C	L	108	6C	l
^M	13	0D	℄	CR	45	2D	-	77	4D	M	109	6D	m
^N	14	0E	℄	SD	46	2E	.	78	4E	N	110	6E	n
^O	15	0F	※	SI	47	2F	/	79	4F	O	111	6F	o
^P	16	10	▶	SLE	48	30	0	80	50	P	112	70	p
^Q	17	11	◀	CS1	49	31	1	81	51	Q	113	71	q
^R	18	12	↕	DC2	50	32	2	82	52	R	114	72	r
^S	19	13	!!	DC3	51	33	3	83	53	S	115	73	s
^T	20	14	¶	DC4	52	34	4	84	54	T	116	74	t
^U	21	15	§	NAK	53	35	5	85	55	U	117	75	u
^V	22	16	▬	SYN	54	36	6	86	56	V	118	76	v
^W	23	17	‡	EIB	55	37	7	87	57	W	119	77	w
^X	24	18	↑	CAN	56	38	8	88	58	X	120	78	x
^Y	25	19	↓	EM	57	39	9	89	59	Y	121	79	y
^Z	26	1A	→	SIB	58	3A	:	90	5A	Z	122	7A	z
^[27	1B	←	ESC	59	3B	;	91	5B	[123	7B	{
^\	28	1C	└	FS	60	3C	<	92	5C	\	124	7C	
]`	29	1D	+	GS	61	3D	=	93	5D]	125	7D	}
^^	30	1E	▲	RS	62	3E	>	94	5E	^	126	7E	~
_	31	1F	▼	US	63	3F	?	95	5F	_	127	7F	Δ [†]

† ASCII code 127 has the code DEL. Under MS-DOCS, this code has the same effect as ASCII 8 (BS). The DEL code can be generated by the CTRL+BKSP key.

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
128	80	Ç	160	A0	À	192	C0	Ļ	224	E0	κ
129	81	ü	161	A1	Á	193	C1	ı	225	E1	ρ
130	82	é	162	A2	Â	194	C2	┆	226	E2	Γ
131	83	â	163	A3	Ã	195	C3	┆	227	E3	Π
132	84	ä	164	A4	Ä	196	C4	—	228	E4	Σ
133	85	à	165	A5	Å	197	C5	+	229	E5	σ
134	86	å	166	A6	Æ	198	C6	┆	230	E6	μ
135	87	ç	167	A7	ß	199	C7	┆	231	E7	γ
136	88	ê	168	A8	¸	200	C8	┆	232	E8	ϕ
137	89	ë	169	A9	¸	201	C9	┆	233	E9	ϑ
138	8A	ì	170	AA	¸	202	CA	┆	234	EA	Ω
139	8B	î	171	AB	½	203	CB	┆	235	EB	δ
140	8C	ï	172	AC	¼	204	CC	┆	236	EC	♠
141	8D	ì	173	AD	¼	205	CD	=	237	ED	♠
142	8E	î	174	AE	¸	206	CE	┆	238	EE	€
143	8F	ï	175	AF	¸	207	CF	┆	239	EF	□
144	90	¸	176	B0	¸	208	D0	┆	240	FO	≡
145	91	¸	177	B1	¸	209	D1	┆	241	F1	+
146	92	¸	178	B2	¸	210	D2	┆	242	F2	>
147	93	¸	179	B3	¸	211	D3	┆	243	F3	<
148	94	¸	180	B4	¸	212	D4	┆	244	F4	∫
149	95	¸	181	B5	¸	213	D5	┆	245	F5	∫
150	96	¸	182	B6	¸	214	D6	┆	246	F6	÷
151	97	¸	183	B7	¸	215	D7	┆	247	F7	÷
152	98	¸	184	B8	¸	216	D8	┆	248	F8	°
153	99	¸	185	B9	¸	217	D9	┆	249	F9	°
154	9A	¸	186	BA	¸	218	DA	┆	250	FA	·
155	9B	¸	187	BB	¸	219	DB	┆	251	FB	·
156	9C	¸	188	BC	¸	220	DC	┆	252	FC	·
157	9D	¸	189	BD	¸	221	DD	┆	253	FD	z
158	9E	¸	190	BE	¸	222	DE	┆	254	FE	■
159	9F	¸	191	BF	¸	223	DF	┆	255	FF	■

Jeu des allumettes

Adaptée du livre de Jacques Deconchat : "102 Programmes Pour MO6, TO8 et TO9+"

Un certain nombre d'allumettes est disposé entre les deux partis, l'ordinateur et vous. Le but du jeu est de ne pas retirer la dernière allumette. Pour se faire, une prise maximale est désignée par le joueur.

En début de partie, on demande :

1. Le nombre d'allumettes disposé entre les deux joueurs (de 3 à 100).
2. Le nombre maximal d'allumettes que l'on peut retirer.
3. Qui commence (0 pour le joueur et 1 pour l'ordinateur).

Puis, tour à tour, chaque parti donne le nombre d'allumettes qu'il prend. L'ordinateur répond grâce à la fonction décrite ci-dessous :

```
int  jeu_ordi (int nb_allum, int prise_max)
{
    int prise = 0;
    int s = 0;
    float t = 0;

    s = prise_max + 1;
    t = ((float) (nb_allum - s)) / (prise_max + 1);

    while (t != floor (t))
    {
        s--;
        t = ((float) (nb_allum-s)) / (prise_max + 1);
    }

    prise = s - 1;
    if (prise == 0)
        prise = 1;

    return (prise);
}
```

La fonction floor donne l'arrondi inférieur d'un nombre.

Nb_allum est le nombre d'allumettes sur la table (au moment du coup à jouer).

Prise_max est le nombre maximum d'allumettes autorisé lors d'une prise.

La fonction retourne en sortie le nombre d'allumettes prises par l'ordinateur.

Exemple :

```
prise = jeu_ordi (10,3) ;
```

La partie se termine lorsqu'il n'y a plus d'allumettes sur la table. La personne ayant tiré la dernière allumette est le perdant, l'autre le vainqueur.

Exemple de solution

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int jeu_ordi (int nb_allum, int prise_max)
{
    int prise = 0;
    int s = 0;
    float t = 0;

    s = prise_max + 1;
    t = ((float) (nb_allum - s)) / (prise_max + 1);

    while (t != floor (t))
    {
        s--;
        t = ((float) (nb_allum-s)) / (prise_max + 1);
    }

    prise = s - 1;
    if (prise == 0)
        prise = 1;

    return (prise);
}

int main ()
{
    int nb_max_d=0;          /*nombre d'allumettes maxi au d,part*/
    int nb_allu_max=0;      /*nombre d'allumettes maxi que l'on peut tirer
au maxi*/
    int qui=0;              /*qui joue? 0=Nous --- 1=PC*/
    int prise=0;            /*nombre d'allumettes prises par le joueur*/
    int nb_allu_rest=0;     /*nombre d'allumettes restantes*/

    clrscr ();

    printf ("Nombre d'allumettes disposées entre les deux joueurs (entre
1 et 300): \n");
    scanf ("%d",&nb_max_d);

    do
    {
        printf ("\nNombre maximal d'allumettes que l'on peut retirer:
\n");
        scanf ("%d",&nb_allu_max);
    }
    while (nb_allu_max >= nb_max_d);
    /* On répète la demande de prise tant que le nombre donné n'est pas
correct */

    printf ("\nQuel joueur commence? 0--> Joueur ; 1--> Ordinateur: \n");
    scanf ("%d",&qui);

    nb_allu_rest = nb_max_d;

    do
    {
```

```
printf ("\nNombre d'allumettes restantes : %d",nb_allu_rest);

if (qui==0)
{
    do
    {
        printf ("\nCombien d'allumettes souhaitez-vous
piocher?\n");
        scanf ("%d",&prise);
    }
    while ((prise > nb_allu_rest) || (prise > nb_allu_max));
    /* On répète la demande de prise tant que le nombre donné
n'est pas correct */
}
else
{
    prise = jeu_ordi (nb_allu_rest , nb_allu_max);
    printf ("\nPrise de l'ordi : %d\n",prise);
}

    qui=!qui;

    nb_allu_rest= nb_allu_rest - prise;
}
while (nb_allu_rest >0);

if (qui == 0)    /* C'est à nous de jouer */
{
    printf ("\nVous avez gagné!\n");
}
else
{
    printf ("\nVous avez perdu!\n");
}

getch ();

return (0);
}
```

Bataille navale simplifiée

Adaptée du livre de Jacques Deconchat : "102 Programmes Pour MO6, TO8 et TO9+"

Le jeu

	0	1	2	3	4	5	6	7	8	9
0	.	.	.	*
1
2
3
4	*
5
6
7
8
9

Ligne ? 6 Colonne ? 3
Distance : 6
Un exemple de Bataille Navale

Nous présentons ici un jeu simplifié de la bataille navale. Celui-ci ne comportera qu'un seul navire et le tableau des cases dans lequel peut se déplacer le navire sera limité à 10 cases sur 10 cases.

A chaque coup on devra donner les coordonnées en donnant l'horizontal (H) et la verticale (V). L'ordinateur répondra si le navire est coulé ou dans le cas contraire donnera la distance qui nous sépare du navire.

En fin de partie, on donnera le nombre de coups nécessaires à la destruction du navire.

Aide pour la programmation

Soit :

d : la distance (int) qui sépare le tir du navire.

x : la position horizontale du navire.

y : la position verticale du navire.

posx_bat : la position horizontale du tir.

posy_bat : la position verticale du tir.

La distance qui sépare le navire du tir est calculé par la formule suivante :

$$d = (\text{int}) (\text{sqrt} ((x - \text{posx_bat}) * (x - \text{posx_bat}) + (y - \text{posy_bat}) * (y - \text{posy_bat})))$$

N'oubliez pas d'inclure la bibliothèque <math.h>

Complément d'informations pour la programmation

En mode texte on dessinera un tableau avec des numéros de 0 à 9 en hauteur et longueur. Les cases en elles-mêmes seront symbolisées par des points.

Dans un premier temps, on indiquera la solution afin de pouvoir contrôler le bon fonctionnement de notre programme.

Dans un second temps, la solution ne sera pas plus donnée et les coups précédemment tirés seront signifiés par des * au lieu du . .

La fonction atoi permet de transformer une chaîne de caractère en nombre. Pour connaître sa syntaxe consulter l'aide.

Exemple de solution

```
#include <conio.h>
#include <math.h>
#include <stdlib.h>

/* Dessine le plateau de jeu */
void plateau ()
{
    int i=0,j=0;

    clrscr ();
    gotoxy (3,1);
    for (i=0; i<10; i++)
        printf ("%d ",i);

    for (i=0; i<10; i++)
    {
        gotoxy (1,2+i);
        printf ("%d ",i);
        for (j=0; j<10; j++)
            printf (". ");
    }

    gotoxy (1,12);
    printf ("Ligne : ");
    gotoxy (11,12);
    printf ("Colonne : ");
    gotoxy (1,13);
    printf ("Distance : ");
}

/* Saisie un chiffre en position x y */
int saisie_chiffre (int x, int y)
{
    char car;
    char chaine [2];
    int sortie = 1; /* Ok on peut sortir */

    do
    {
        /* On efface la précédente case */
        gotoxy (x,y);
        printf (" ");

        /* On se repositionne pour la saisie */
        gotoxy (x,y);
        car = getch ();
        if ((car<'0') || (car>'9'))
        {
            /* Saisie incorrect : beep */
            sound (1000);
            delay (10);
            nosound ();
            /* On doit recommencer */
            sortie = 0;
        }
        else
            sortie = 1;
    }
}
```

```
while (!sortie);

/* Affichage du caractère saisi */
gotoxy (x,y);
printf ("%c",car);

/* atoi transforme une chaîne de caractère en nombre */
/* transformation de notre caractère en chaîne */
chaîne [0] = car;
chaîne [1] = 0;
return (atoi (chaîne));
}

void main ()
{
    int posx_bat=0, posy_bat=0; /* Position du bateau */
    int x,y; /* Position du tir */
    int nb_coups=0; /* Nombre de coups */
    int d=0; /* Distance */

    randomize ();
    posx_bat = random (10); /* 0 ... 9 */
    posy_bat = random (10);

    plateau ();
    do
    {
        /* Saisie de la colonne */
        x=saisie_chiffre (9,12);

        /* Saisie de la ligne */
        y=saisie_chiffre (21,12);

        /* Nombre de coups */
        nb_coups ++;

        /* Affichage du coup */
        gotoxy ((x+1)*2+1, y+2);
        printf ("");

        /* Effacement des coordonnées */
        gotoxy (9,12);
        printf (" ");
        gotoxy (21,12);
        printf (" ");

        /* Calcul de la distance */
        d = (int) (sqrt (((x-posx_bat)*(x-posx_bat))+((y-posy_bat)*(y-
posy_bat))));

        /* Affichage de la distance */
        gotoxy (12,13);
        printf ("%d",d);
    }
    while (!(x==posx_bat) && (y==posy_bat));
/* On recommence tant que la position saisie n'est pas celle du navire */

    gotoxy (1,15);
    printf ("Nombre de coups : %d",nb_coups);
    getch ();
}
```