

|  |  |
| --- | --- |
| **SYSTEME D’INFORMATION NUMERIQUE**  **Activité expérimentale** | |
| **Term** | **Langage C**  **Codage et Traitement de l’information** |

|  |  |  |  |
| --- | --- | --- | --- |
| **DONNEES PEDAGOGIQUES** | | | |
| Centres d’intérêts : Approche fonctionnelle et structurelle d’une chaîne d’information | | | | |
| Support | Compétences attendues | Connaissances abordées | | |
|  | **CO2.1.** Décoder le cahier des charges d’un produit, participer, si besoin, à sa modification.  **CO3.1.** Identifier et caractériser les fonctions et les constituants d’un produit ainsi que ses entrées/sorties  **CO3.2.** Identifier et caractériser l’agencement matériel et/ou logiciel d’un produit. | 2.4.3.Codage et traitement de l’information  4.3.5. Conception informationnelle des produits | | |
| Type d'activité | Activité expérimentale |  | | |
| Prérequis | Apport de savoir, Cours : « Opérateurs Logiques » | |  | |
| Durée | 6h00 |  | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Première | ◼◼◼◼◼◼◼ | ◼◼◼◼◼◼◼ | ◼◼◼◼◼◼ | ◼◼◼◼◼◼ | ◼◼◼◼◼◼ |
| Terminale | ◼◼◼◼◼◼◼ | ◼◼◼◼◼◼◼ | ◼◼◼◼◼◼ | ◼◼◼◼◼◼ | ◼◼◼◼◼◼ |

|  |  |
| --- | --- |
| **DONNéES TECHNIQUES** | |
| Matériel | PC | |
| Documentation | Le présent document | |
| Logiciel | CodeBlocks ou Dev C++ | |

Contexte et Histoire Sources Wikipédia

Le langage C a été inventé au cours de l'année 1972 dans les Laboratoires Bell. Il était développé en même temps que le système d’exploitation UNIX par Dennis Ritchie et Ken Thompson. Ken Thompson avait développé un prédécesseur de C, le langage B, qui est lui-même inspiré de BCPL (Basic CombinedProgrammingLanguage). Dennis Ritchie a fait évoluer le langage B dans une nouvelle version suffisamment différente, en ajoutant notamment les types, pour qu'elle soit appelée C.

Bien que C soit officiellement inspiré de B et de BCPL, on note une forte influence de PL/I (ou de PL360, Programming Language Number 1) ; on a pu dire que le langage C était à Unix et au PDP-11 ce que PL/I fut pour la réécriture de Multics (Système d’exploitation en temps partagé).

Par la suite, Brian Kernighan aida à populariser le langage C. Il procéda aussi à quelques modifications de dernière minute.

En 1978, Kernighan fut le principal auteur du livre *The C ProgrammingLanguage* décrivant le langage enfin stabilisé ; Ritchie s'était occupé des appendices et des exemples avec Unix. On appelle aussi ce livre « le K&R », et l'on parle de **C traditionnel** ou de **C K&R** lorsqu'on se réfère au langage tel qu'il existait à cette époque.

En 1983, l'Institut national américain de normalisation (ANSI) a formé un comité de normalisation (X3J11) du langage qui a abouti en 1989 à la norme dite **ANSI C** ou **C89** (formellement ANSI X3.159-1989). En 1990, cette norme a également été adoptée par l'Organisation internationale de normalisation (**C90**, **C ISO**, formellement ISO/CEI 9899 :1990). ANSI Cest une évolution du C K&R qui reste extrêmement compatible. Elle reprend quelques idées de [C++](https://fr.wikipedia.org/wiki/C%2B%2B), notamment la notion de prototype et les qualificateurs de type.

En 1999, une nouvelle évolution du langage est normalisée par l'[ISO](https://fr.wikipedia.org/wiki/Organisation_internationale_de_normalisation) : **C99** (formellement ISO/CEI 9899:1999). Les nouveautés portent notamment sur les tableaux de taille variable, les pointeurs restreints, les nombres complexes, les littéraux composés, les déclarations mélangées avec les instructions, les fonctionsinline, le support avancé des nombres flottants, et la syntaxe de commentaire de C++. La bibliothèque standard du C99 a été enrichie de six fichiers d'en-tête depuis la précédente norme.

D’autres évolutions des normes ont été réalisées. Je vous laisse le soin de consulter cela sur le net : <https://fr.wikipedia.org/wiki/C_(langage)>

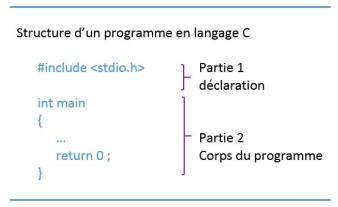
Le langage C est utilisé dans :

* Le développement de systèmes embarqués ;
* Les jeux vidéo (plus trop dans le monde professionnel comparé au **C**++) ;
* La programmation système ;
* Le développement d'applications temps réel ;
* Les bibliothèques de fonctions
* …

Structure d’un programme en langage C

Tout programme C doit comporter une fonction principale « **main**». Cette fonction est celle utilisée par le système pour exécuter le programme.

#include <stdio.h>

int main(int argc, char \*argv[])

{

printf("hello, world\n") ;

   return 0 ;

}

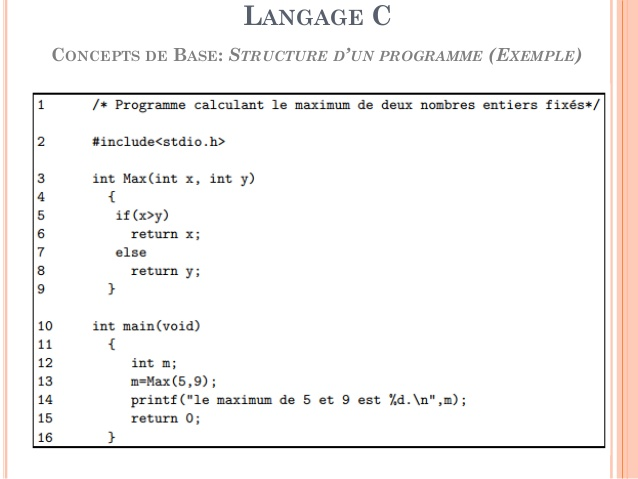
La partie déclaration comprend :

* Les « includes » : inclusion des bibliothèques du langage C, ou de fonctions spécifiques
* Les « define » : définition des constantes
* Les variables globales

Le corps du programme est composé :

* De la fonction principale « main »
* Des fonctions appelées par la fonction « main »

Exemple :



Scope

Retour de la fonction Max

Les principaux types des variables

|  |  |  |
| --- | --- | --- |
| **Types entiers, en ordre croissant** | | |
| **Type** | **Taille** | **Magnitude *minimale* exigée par le standard** |
| **char** | -128 à 127, ou 0 à 255, selon l'implémentation | 8 bits |
| **signed char** | -128 à 127 |
| **unsigned char** | 0 à 255 |
| **short signed short** | -32 768 à +32 767 | 16 bits |
| **unsigned short** | 0 à 65 535 |
| **int signed int** | -32 768 à +32 767 | 16 bits |
| **unsigned int** | 0 à 65 535 |
| **long signed long** | -2 147 483 648 à +2 147 483 647 | 32 bits |
| **unsigned long** | 0 à 4 294 967 295 |
| **float (simple précision)** | -1,2 ×10-38 à 3,4 ×10+38 |
| **double (double precision)** | -1.7\*10-308 à 1.7\*10308 | 64 bits |
| **long long (C99) signed long long (C99)** | -9 223 372 036 854 775 808 à +9 223 372 036 854 775 807 |
| **unsigned long long** | 0 à 18 446 744 073 709 551 615 |

Recherches et documentations

***Qu'est-ce que le langage C:***Grâce au lien <https://fr.wikipedia.org/wiki/C_(langage)>, faire une lecture de la présentation, puis résumer ce que vous avez appris ci-dessous.

|  |
| --- |
|  |

Partie 1 : Affichage d’un texte sur un écran console

Ouvrir le logiciel « CodeBlocks » et créer un nouveau projet de type console.

Copier la séquence de programme ci-dessous, puis, après l’avoir enregistrée, la compiler puis l’exécuter.

#include <stdio.h>

int main ()

{

printf ("Bonjour");

getchar ();

return 0;

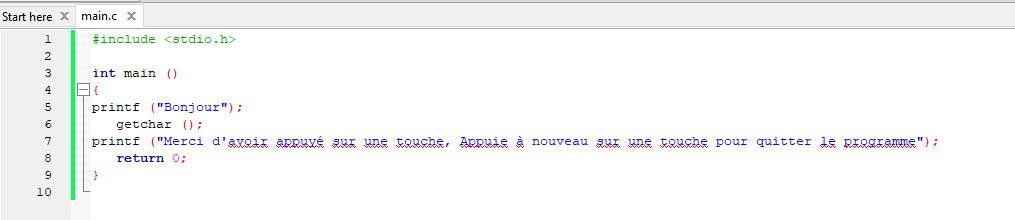
}

1. Que fait la fonction « getchar() »

C'est une fonction sans paramètre qui « capte » la valeur du caractère lu à la position courante de l'entrée standard

1. Écrire le programme permettant de réaliser l’algorithme ci-dessous

* Affiche « Salut à toi, appuie sur une touche s'il te plaît » ;
* Attend l'appui d'une touche ;
* Affiche : « Merci d'avoir appuyé sur une touche, Appuie à nouveau sur une touche pour quitter le programme » ;
* Attend l'appui d'une touche.

******

**FAIRE VALIDER VOTRE TRAVAIL PAR LE PROFESSEUR**

Partie 2 : Utilisation de variables

1. Premier exemple avec des variables du type entier. Pour cela, ouvrir le logiciel « CodeBlocks » et créer un nouveau projet de type console.

Copier la séquence de programme ci-dessous, puis, après l’avoir enregistrée, la compiler puis l’exécuter.

#include <stdio.h>

int main ()

{

int i;

int j;

i=22;

j=i;

printf ("i vaut %d\n", i);

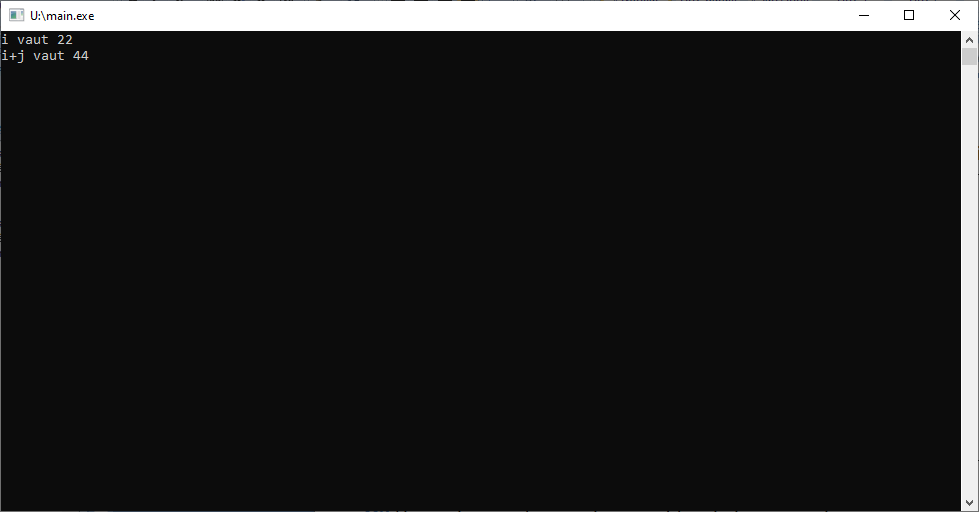
printf ("i+j vaut %d\n", i+j); \*/

getchar ();

return 0;

}

Insérer la copie d’écran de la console.



1. Second exempleavec des variables du type caractère.

#include <stdio.h>

int main ()

{

char car;

char f;

car='E';

f='e';

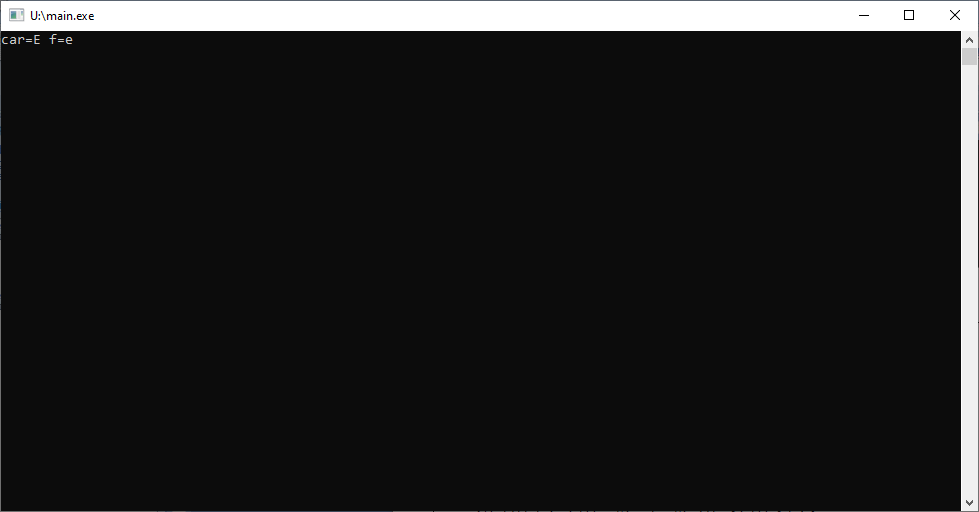
printf("car=%c f=%c\n",car,f);

getchar ();

return 0;

}

**Insérer la copie d’écran de la console.**

****

1. Utilisation de % dans printf

À l'intérieur du premier paramètre d'un printf (appelé le format), l'emploi de « %x » signifie qu'à l'exécution, %x doit être remplacé par la valeur correspondante qui figure dans les paramètres suivants, après transformation de ce paramètre dans le type puis le format désigné par x. Nous avons à notre disposition plusieurs formats d'affichage,comme : %d, %c, etc…

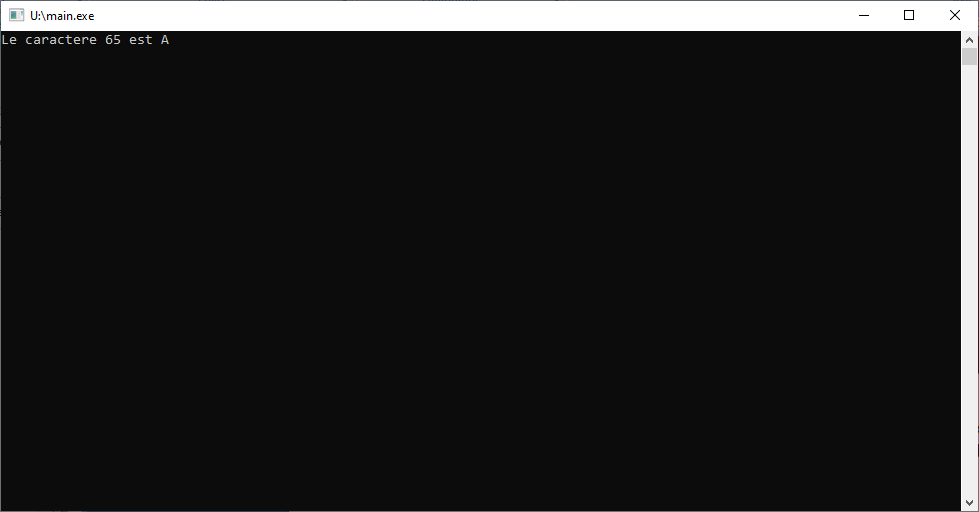
Exemple à tester dans un programme complet.

int i;

i=65;

printf ("Le caractère %d est %c",i,i);

Insérer le résultat obtenu, puis commentez ce résultat.



**FAIRE VALIDER VOTRE TRAVAIL PAR LE PROFESSEUR**

1. Déclarez des variables avec les valeurs suivantes 70, 82, 185 et 30 puis affichez le contenu de ces variables.

70 = F ; 82 = R; 185 = ╣; 30 = ;

1. Faites la même chose avec les caractères c, o, u, C, O, U.

c = 99 ; o = 111 ; u = 117 ; C = 67 ; O = 79 ; U = 85 ;

**FAIRE VALIDER VOTRE TRAVAIL PAR LE PROFESSEUR**

Partie 3: Faire des calculs en langage C

1. Introduction à une calculatrice

Écrivez 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 : »

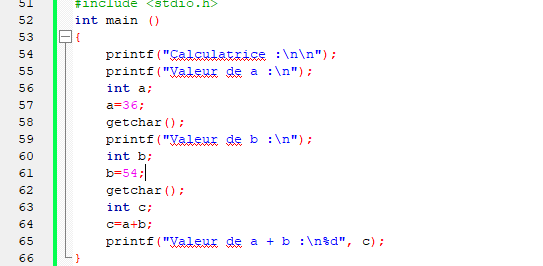
**Pour l’instant votre programme n’affiche que du texte !!! Maintenant il faut qu’il fasse les calculs.**

Pour pouvoir faire des calculs, complétez le programme en :

• Déclarant 2 variables a et b de type « int » (entier) ;

• Assignant à ces deux variables les valeurs 36 et 54 ;

• Faisant afficher le résultat de la somme de a+b **(attention, n'écrivez pas le résultat 90 dans votreprogramme, mais faites bien faire le calcul par l’ordinateur !!!!).**



1. Saisie des variables

Si une calculatrice électronique se limitait à calculer la somme de deux nombres fixes que l’on ne peut pas changer, le boulier serait encore très répandu.

Pour donner une valeur à une variable en utilisant le clavier, il est possible d'utiliser la fonction « scanf ».

La fonction « scanf » s'utilise de la façon suivante :scanf ("%d", &a); // saisie de la valeur a

Comme pour « printf », nous reconnaissons le « %d » pour la saisie d'un nombre entier. Le « & » devant le « a » signifie que nous allons écrire dans la variable a.

En fait « &a » signifie « l'adresse mémoire de a ». La fonction « scanf » va donc écrire dans l'emplacement mémoire.

Nous allons maintenant saisir les variables a et b,dans le programme précédent.

Pour exemple, voici le code pour la saisie de « a », **la saisie de b reste à faire par vos soins à titre d'exercice**.

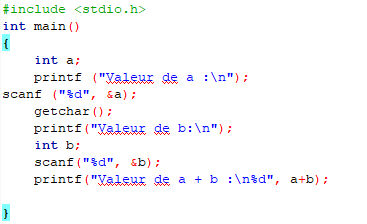
/\* Saisie de la valeur de a \*/

printf ("Valeur de a :\n");

scanf ("%d", &a);

Initialisation des variables : Il est conseillé d'initialiser les variables avant de les utiliser. Au début du programme, après leur déclaration, assignez la valeur 0 à a et à b.Une fois complété, compilé, nous allons tester votre programme.

Entrez une valeur pour a puis appuyez sur la touche ENTREE. Renouvelez ensuite l'opération pour donner la valeur de b. Vérifiez que le résultat est correct.



**FAIRE VALIDER VOTRE TRAVAIL PAR LE PROFESSEUR**

1. Déclarez une troisième valeur de type int (pensez à l'initialiser à 0) que nous nommerons simplement s comme somme. Une fois les valeurs de a et b saisies, initialisez s avec la valeur de a+b. Affichez la valeur de s. Nous devrions avoir les mêmes résultats qu'auparavant, bien sûr.
2. Réalisez deux petits programmes qui font :

* La soustraction de deux nombres ;
* La multiplication de deux nombres.
* La division de deux nombres.

Testez le fonctionnement de votre programme, puis prendre :

* Pour le 1er nombre 5 et pour le 2nd 2, analyser et justifier le résultat
* Pour le 1er nombre 2 et pour le 2nd 5, analyser et justifier le résultat

**FAIRE VALIDER VOTRE TRAVAIL PAR LE PROFESSEUR**

Partie 4: Les types flottants

Nous allons étudier un nouveau type de données : les nombres à virgule flottante ou simplement flottants (« float »), qui permettent de représenter des nombres à virgule. Le type « float » permet de déclarer un tel nombre. Transformez les trois programmes précédents en utilisant le type « float » au lieu du type « int ».

Enfin, si pour les « int », nous utilisions le format « %d » au sein des « printf » et des « scanf », à présent, nous allons utiliser le format « %f » pour les flottants.

Pour vous aider, voici un petit morceau de programme qui permet la saisie de la valeur de a et l'affiche :

printf("Saisie de a :");

scanf("%f",&a);

printf("\n a vaut : %f\n",a);

Pour un affichage plus agréable il est possible de fixer le nombre de chiffres après la virgule dela façon suivante :

« %.[nombre de chiffres après la virgule]f »

Voici un exemple :

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

**FAIRE VALIDER VOTRE TRAVAIL PAR LE PROFESSEUR**

1. Créez un ultime programme qui réalise la division de deux nombres à virgules. Vous pourrez vous amuser à le tester avecune division par 0 ! La solution à ce petit problème sera vue dans le chapitre sur les conditions (« if »).

**FAIRE VALIDER VOTRE TRAVAIL PAR LE PROFESSEUR**

Partie 5: Premiers programmes utilisant des boucles et des conditions.

Exemple d’une boucle « si alors sinon » : Tester ce programme et ajouter des commentaires qui expliquent le rôle de chaque ligne et non de chaque instruction !

1. **Etape 1 :*Si…. Alors….Sinon***

#include <stdio.h>

int main ()

{

int valeur1; // val 1

int valeur2; // val 2

/\* Saisie de valeur1 \*/

printf ("Entrez une 1ere valeur : ");

scanf ("%d",&valeur1);

/\* Saisie de valeur2 \*/

printf ("Entrez 2eme valeur : ");

scanf ("%d",&valeur2);

if (valeur1<valeur2)// si val1 < val2

printf("La plus grande valeur est: %d\n",valeur2);//imprimer

else //sinon

printf("La plus grande valeur est: %d\n",valeur1); //imprimer

return 0; //retourne rien

}

**FAIRE VALIDER VOTRE TRAVAIL PAR LE PROFESSEUR**

1. Exemple d’un programme utilisant des opérateurs logiques (booléens): Tester ce programme et ajouter des commentaires pour chaque ligne. **(Le programme est incomplet, ajouter le nécessaire).**

Pour connaitre la signification de ces opérateurs, consultez Internet (<https://web.maths.unsw.edu.au/~lafaye/CCM/c/cop.htm>).

int i1=1;

int i2=0;

printf("i1 || i2 = %d",i1||i2);

/\* affichera 1 car : vrai||faux=vrai et vrai vaut 1 \*/

printf("i1 && i2 = %d",i1&&i2);

/\* affichera 0 car : vrai&&faux=faux et faux vaut 0 \*/

printf("contraire(1)=%d",!(1));

/\* affichera 0 car : !(vrai)=faux et faux vaut 0 \*/

**FAIRE VALIDER VOTRE TRAVAIL PAR LE PROFESSEUR**

1. Variable positive, négative ou nulle

Faites saisir une variable de type entier et indiquez à l'utilisateur si celle-ci est strictement positive, strictementnégative ou nulle.

Votre code contiendra quelque chose comme ceci :

if (a > 0)

printf ("Valeur positive");

else

printf ("Valeur négative");

**FAIRE VALIDER VOTRE TRAVAIL PAR LE PROFESSEUR**

**A RETENIR :**

• La valeur Vrai peut être assimilée à la valeur numérique 1 ou à toute valeur non nulle.

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

• Ne pas oublier les parenthèses lorsqu'il y a un if :

if a > 0 // ne sera pas compilé !!!

printf ("Valeur positive");

else

printf ("Valeur négative");

• au contraire, il faudrait écrire :

if (a > 0)

printf ("Valeur positive");

else

printf ("Valeur négative");

• Différencier l'opérateur d'affectation =

• Et l'opérateur de comparaison ==.

1. **Etape 2 : *Faire Tant Que***

« Do » … « while » (traduisez par Faire Tant que) permet de réaliser une suite d'instructions tant qu'une ou plusieurs conditions sont remplies.

**1ère exemple :**

Saisir, testez, commentez le programme ci-dessous, puis lisez les explications qui figurent en dessous du programme.

#include <stdio.h>

int main()

{

char car=' ';

int sortie=0;

do {

printf ("Appuyez sur S pour sortir !\n");

car = getchar ();

/\* On le compare pour savoir si l'on peut sortir: \*/

sortie = ((car == 's') || (car == 'S'));

}

while (sortie==0);

return 0;

}

**Quelques remarques :**

Notez que la touche ENTREE utilisée pour la saisie du caractère est-elle même traitée comme un caractère entrée et provoque donc aussi l'affichage de la phrase « Appuyez sur S pour sortir… » qui s'affiche donc deux fois.

**• un nombre entier vaut la valeur logique vraie si ce nombre est différent de 0.**

**• un nombre entier vaut la valeur logique faux si ce nombre est égal à 0.**

**• || est un « 'ou' logique », donc au niveau de la ligne :**

**2nd exemple :**Programme qui boucle et ne s’arrête pas,

Pour stopper un programme qui boucle, il faut presser simultanément sur les touches CTRL+C (break).

Vous pouvez tester cette possibilité en exécutant le programme suivant :

#include <stdio.h>

int main () {

int i=1;

do {

printf(" i=%d \n",i);

i=i+1;

}

while (i>0); /\* Test toujours vrai ! \*/

return 0;

}

**Exercice :** Sortir de la boucle

Transformez l'exemple précédent afin que l'on sorte de la boucle uniquement quand l'utilisateur a saisi le nombre 10.

Question préliminaire : Rappeler comment on saisit un nombre.

1. **Etape 3 : *Itération de A à B au pas***

***for ( X = Xi ; X <= Xf ; X+pas)*** permet de réaliser une suite d'instructions un certain nombre de fois ***(Xf –Xi)*** au rythme de ***pas***

**Programme de base :**Saisir, testez, puis commentez le programme ci-dessous,

#include <stdio.h>

int main ()

{

int i; int j;

for (i=1; i<=5; i++)

{

for (j=1; j<=i; j++) printf ("\*");

}

printf ("\n");

getchar();

return 0;

}

**Exercice :**Modifiez le programme précédent pour obtenir la représentation ci-dessous

|  |  |
| --- | --- |
|  | • sur la ligne n° 1, afficher 9 espaces puis 1 étoile;  • sur la ligne n° 2, afficher 8 espaces puis 3 étoiles;  • sur la ligne n° 3, afficher 7 espaces puis 5 étoiles;  • ……  • sur la ligne n° i, afficher 10-i espaces puis 2\*i-1 étoiles. |

Aller, je vous aide un peu, le programme à utiliser est proche de celui-ci-dessous.

|  |  |
| --- | --- |
| for (i=1; i<=10; i++) {  for (j= **?**;j< **?**;j++) // les espaces...  printf(" ");  for (j=**?**; j<(**?**); j++)  printf("\*");  printf("\n"); |  |

➀ Comprendre, puis commenter cette séquence de programme

➁ Trouver les valeurs pour remplacer les points d’interrogation

➂Insérer cette séquence dans le programme précédent tester le.

**FAIRE VALIDER VOTRE TRAVAIL PAR LE PROFESSEUR**

**Exercice:**Rajoutez le tronc maintenant

|  |  |
| --- | --- |
|  | Rajouter la séquence d’instructions permettant de doter notre sapin d’un tronc.  Vous avez réussi ? , il ne manque plus que les guirlandes …. ! |

**FAIRE VALIDER VOTRE TRAVAIL PAR LE PROFESSEUR**

Partie 6: Les compteurs.

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.

Par exemple:

int i;

i=1;

i=i+1;

printf("i vaut: %d",i); // affichera 2

**Commentaires :**Lorsque nous écrivons i=1, l'ordinateur range la valeur 1 dans la case mémoire désignée par i. Lorsque nous demandons : i=i+1, l'ordinateur commence par prendre connaissance de la valeur de i en mémoire (ici 1) et lui ajoute 1. Le résultat est stocké dans la case mémoire associée à i. Finalement, i vaudra 2.

Pour gagner du temps, le langage C nous permet de remplacer une expression comme : i=i+1 par l'expression suivante : i++ qui fera exactement la même chose.

int i;

i++; /\* Incrémente le compteur i \*/

i--; /\* Décrémente le compteur i \*/

**Exemple :**Saisir, testez, commentez le programme ci-dessous, puis lisez les explications qui figurent en dessous du programme.

#include <stdio.h>

int main ()

{

int nbre = 0;

int nb\_nbre = 0;

printf("Tapez 10 nombres pour arrêter ...");

do {

scanf("%d",&nbre);

nb\_nbre ++;

}

while (nb\_nbre != 10);

return 0;

}

**Exercice:** Saisi de 10 caractères

Écrivez le programme : Tant que je n'ai pas saisi 10 caractères, je recommence la saisie d'un caractère.

**A retenir :**Voici un exemple de programme ci-dessous qui résume ce qui a été vu dans ce chapitre. Ce programme doit afficher à l'écran tous les nombres pairs inférieurs à 100 :

|  |  |
| --- | --- |
| 1èreVersion : | 2ème Version |
| #include <stdio.h>  int main () {  int i = 0;  while (i!=100) {  if (i%2==0) /\* reste de la division de i par 2 \*/  printf("%d ",i);  /\* pas de else ni de {} ici, c'est inutile...\*/  i++;  }  return 0;  } | #include <stdio.h>  int main () {  int i = 0;  while (i!=100) {  printf("%d ",i);  i+=2;  }  return 0;  } |

**A retenir (suite):**Ces 2 programmes utilisent l’opérateur modulo. C’est l'opérateur qui donne le reste de la division entière il est noté % en C. Ainsi, 10%2 vaut 0 car la division entière de 10 par 2 donne 5 et il n'y a pas de reste. En revanche, 10%3 vaut 1 car la division entière de 10 par 3 donne 3 et il reste 1.

Tracer maintenant dans votre compte rendu les algorigrammes des 2 programmes ci-dessus.

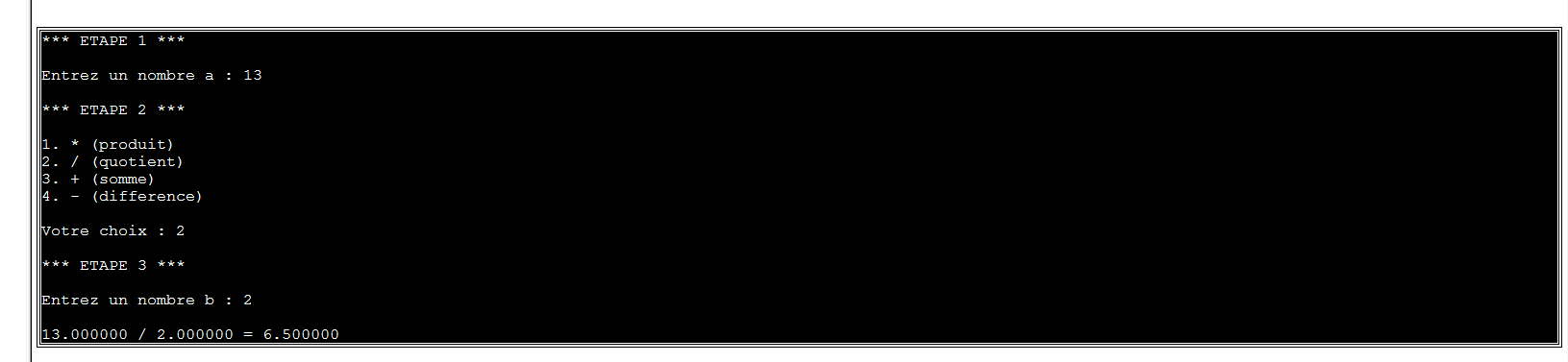
Partie 7: Fonctions et appel de fonctions

**Exercice :** Calculatrice, la suite

A l’issue du cours sur les fonctions,

➀ Créer 4 fonctions mathématiques de base (somme, différence, produit et quotient)

➁ Réaliser le programme *Calculatrice V2* donnant le résultat ci-dessous



**Exercice 9 : La loi d’Ohm**

Ecrire un programme qui calcul la valeur de la tension(U) aux bornes d’une résistance(R) parcourue par un courant (I). Utiliser la loi d’Ohm.

Consignes :Les variables du programme sont U, R, I.Déclarer une fonction et un programme principal.

Le programme principal invite l’utilisateur à saisir la valeur de R et I et affiche la valeur de la tension U

**Exercice 10 : La loi d’Ohm**

On reprend le même problème que l’exercice précédent avec des fonctionnalités supplémentaires (calcul du courant, calcul de la tension, calcul de la résistance).

Consignes :Le programme va proposer à l’utilisateur trois choix :

1. Pour calculer la tension taper 1

2. Pour calculer le courant taper 2

3. Pour calculer la résistance taper 3

**Exercice 11 : La synthèse**

Regrouper dans un programme une page d’accueil qui permet de choisir entre les 4 opérations calculatrices, les opérations de l’exercice 3 et les calculs circonférence et surface d’un cercle.