Parcours GESE

Semestre 1

Algorithme et programmation 1

Manuel de Travaux Pratiques

Préparé par :

Prof. Ghita Mangoub

A.U : 2025-2026

Travaux pratiques numéro 1
Initiation

I- but

Ce TP a pour objet:

1) La familiarisation avec l'environnement du C.
2) Manipulation des entiers et des réels, affectation, expression, saisie et affichage
3) apprendre à faire quelques manipulation sur des variables : permutation, équivalent ASCII, …etc et en général se familiariser d'avantage avec la structure séquentielle (linéaire).

II- Quelques notions à savoir

Eléments de base du langage C

1- Les commentaires non affichable

Possibilité d'introduire des commentaires. On utilise alors le

Signe // s'il s'agit d'un commentaire sur une seule ligne.

Pour le cas d'un commentaire qui porte sur plusieurs lignes, on le commence par le signe /* et on le ferme par le signe */.

Exemple1

```
// Ceci est un commentaire sur une seule ligne
```

Exemple2

```
/* Ceci est un autre
Commentaire
Sur plusieurs lignes
*/
```

2- Types de bases

3 types élémentaires :

Les entiers

Courts(short int), normaux(int), long(long int)

Exemple :

```
int a ;
```

short int a, b ;

Les réels

Courts(float)(32 bits)(6 chiffres significatifs), long(double)(64 bits)(16 chiffres),très long(long double)(80 bits)

Exemple :

float a ;
double a, b ;

Les caractères

Char

Exemple : char a,b,c ;

B-2 constantes, variables

Un programme traite une ou des variables entrantes : les données

Il restitue une ou des variables sortantes : les résultats

Un programme qui ne satisfait pas à ces deux conditions n'a pas de justification et par conséquent n'existe pas. Les données traitées par le programme sont de deux types : variable ou constante. Chaque variable et constante doit être déclarée avant d'être utilisé : nom + type. Cette déclaration consiste en une réservation en mémoire de la place nécessaire et suffisante à l'hébergement de la valeur.

2-1) Notion de constante : une constante porte un nom qu'on appelle identificateur et garde la même valeur On la déclare de la façon suivante :

const type identificateur = valeur_de_la_constante ;

Exemple :

Const int N = 10 ;

C'est à dire qu'on déclare que N est une constante entière qui a pour valeur 10.

On peut aussi déclarer une constante de la manière suivante :

#define nom valeur ;

Exemple : #define N 10

2-2) Notion de variable : Une variable est un endroit en mémoire permettant de stocker une valeur. Une variable est caractérisée par un nom appelé un identificateur, par un type (l'ensemble dans lequel elle prend ses valeurs) et par une valeur.

Exemple : si notre variable est la masse alors pour identificateur on peut choisir m, pour le type c'est un réel et pour la valeur 30Kg.

La syntaxe de la déclaration est la suivante :

Type identificateur ;

Exemple1 : int a ;

Pour déclarer la variable dont le nom est l'identificateur a comme entier qui peut être positif ou négatif.

Exemple2 : double X,Y,Z ;

Pour déclarer les 3 variables dont les noms respectifs sont les identificateurs X,Y et Z comme réels qui peuvent être positifs ou négatifs.

Exemple3 : bool c ;

Pour déclarer la variable dont le nom est l'identificateur c comme boeliène. C'est à dire une expression logique qui ne peut avoir que deux valeurs soit vrai soit faux.

Exemple4 : char a,b,c ;

Pour déclarer les 3 variables dont les noms respectifs sont les identificateurs a, b et c comme caractères.

En ce qui concerne l'identificateur, il peut être constitué de plusieurs lettres, chiffres et mots dans ce derniers cas les différents mots sont séparés par un signe souligné c'est à dire « _ «

Exemple : int X_a ;

Float Y_b ;

Quelques notions utiles:

1) L'identificateur ne doit pas commencer par un chiffre. Il doit être en rapport avec ce que contient la variable. Il ne doit pas contenir d'espaces ni contenir le caractère -. En général un identificateur doit refléter la variable qu'il contient

Exemple : si notre variable est la masse d'un premier objet, on peut choisir comme identificateur : masse_objet1

2) On peut copier directement une variable initialement déclarée comme int dans une variable initialement déclarée comme double.

Exemple int a ;
double X ;
X=a ;
Alors que pour copier une variable initialement déclarée comme double dans une variable initialement déclarée comme int, il faut d'abord transformer la variable réelle en une variable entière .
Exemple int a ;
double X ;
a=(int)X;
3) Tous les caractères sont repérés dans un système de représentation appelé le code ASCII. Dans ce code, à chaque caractère correspond un entier allant de 0 à 127. En C++, pour récupérer le code ASCII d'un caractère donné, il suffit de mettre le caractère dans un entier de la manière suivante :
Char a ; int x ;
On lit le caractère en question, puis on le met dans l'entier x de la façon suivante : x=a ;
Donc on peut comprendre que lorsqu'on compare des caractères, c'est en fait leur code ASCII qu'on compare.
4) On peut aussi transformer un caractère qui est une lettre majuscule (minuscule) en un caractère qui est une lettre minuscule(majuscule) pour cela, il suffit de lui ajouter 'a'-'A' ('A'-'a') de la manière suivante :
Char a, b ;
On lit une lettre majuscule et on la place dans a
On transforme cette lettre en celle qui lui correspond en minuscule en plaçant le résultat dans b de la manière suivante :
B= a+('a' – 'A') ;
1) Si on a un caractère qui correspond à l'un des chiffres allant de 0 à 9, on peut transformer ce caractère en un entier de la manière suivante :
Char a ; int x ;
On lit le caractère a

Si a>='0' et a<=9 alors on le transforme en son chiffre correspondant de la manière suivante : x=a – '0' ;

3- Opération entré – sortie

```
printf("A = ");
 scanf("%lf", &A);
```

B-4 Expressions

Une expression peut être une instruction ou une suite d'instructions séparées par l'un des opérateurs suivants :

Opérateurs d'assignation

Affectation d'une valeur ou d'une expression x à une variable y : (y = x) Addition de la valeur de gauche à la valeur de droite : + = (x = x+2)
Soustraction de la valeur de gauche à la valeur de droite : - =
Multiplication de la valeur de gauche à la valeur de droite : * =
Division de la valeur de gauche à la valeur de droite : / =
Incrémentation : ++
Décrémentation : - -

Opérateurs comparateurs de nombres

Egalité : = =
Infériorité stricte : <
Supériorité stricte : > Infériorité : < =
Supériorité :> =
Différent : ! =

Opérateur logiques

OU : ||

ET : &&

NON :!

Travail à effectuer

EX1_TP1

Taper compiler et exécuter un programme sous C qui lit trois variables **entières** A,B et C et qui calcule : A+B, B*C, A/C et A%(B+C)
Exécutez le programme pour les cas suivants :
1) C=0
2) C=B=0
3) C et B différents de zéro

EX2_TP1

Ecrire un programme qui fait la permutation circulaire de quatre nombres lus au clavier.

EX3_TP1

Taper compiler et exécuter un programme sous C qui donne le code ASCII de certains caractères déclarés constants dans le programme (?, y et U par exemple).

Solution

EX1_TP1

```
//utilisation de quelques opérations avec des entiers
#include<stdio.h>
#include<stdlib.h>
int main()
{
//declaration
  int A,B,C,X,Y,Z,W;

//preparation du traitement
printf("donner trois entiers A,B et C\n");
scanf("%d%d%d",&A,&B,&C);
//traitement
X=A+B ;
Y= B*C ;
Z= A/C ;
```

```
W =A%(B+C) ;
    {printf("X= %d \n",X);}
{printf("Y= %d \n",Y);}
{printf("Z= %d \n",Z);}
{printf("W= %d \n",W);}
printf("FIN \n");
}
```

EX2_TP1

```
/*Ecrire un programme qui fait la permutation circulaire
de quatre nombres lus au clavier. */
#include<stdlib.h>
#include<string.h>
int main()
{
//declaration
  int A,B,C,D,ECH;

//preparation du traitement
printf("Donner quatres entiers A,B,C et D \n");
scanf("%d%d%d%d",&A,&B,&C,&D);
ECH=A; A=D; D=C; C=B; B=ECH;
printf("Apres permutation on a: \n");
printf("A=%d et B=%d et C=%d et D=%d \n",A,B,C,D);
}
```

EX3_TP1

```
// EX4_TP2
#include <stdio.h>

#include <stdlib.h>

//Déclaration

int main()

{

const char x=' ?', y='y', z='U';


int X,Y,Z;

// traitement

//Edition des résultats
```

```
printf("le code ASCII du caractere %c est: %d\n",x, X);
printf("le code ASCII du caractere %c est: %d\n",y, Y);
printf("le code ASCII du caractere %c est: %d\n",z, Z);
 printf("FIN \n");
 }
```

Travaux pratiques numéro 2
Structures de sélection

I- but

Dans ce TP, il s'agit d'apprendre à utiliser la structure de sélection appelée aussi structure alternative avec ses différentes variantes.

II- Quelques notions à savoir

A- Les différents cas de la sélection

A-1) Sélection avec une seule alternative (structure alternative simple)

Lorsqu'il y'a une seule condition à vérifier, la syntaxe en C se présente de la manière suivante.

if **(**condition) **{** traitement ;}

A-2) Sélection avec deux alternatives

Lorsqu'il y'a une deux condition à vérifier, la syntaxe algorithmique peut se présenter deux manières.

A-2-1) Première variante (structure alternative simple répétée)

if(Condition1) {traitement1 ;}
if (Condition2) {traitement2 ;}

A-2-2) Deuxième variante (structure alternative complète)

if (Condition1) {traitement1 ; }
else {traitement2 ;}

A-3 Imbrication de la sélection (la structure alternative complète imbriquée)

Elle est aussi utilisée lorsqu'il y' a N conditions à vérifier.

```
                 if  (condition1)     { traitement1 ;}
else          if  (condition2)     { traitement2 ;}
else          if  (condition3)     { traitement3;}
                                    .
                                    .
else          if  (conditionN-1)   { traitementN-1 ;}
else                               { traitementN;}
```

A-4 Le choix multiple (la structure alternative de cas)

Elle est aussi utilisée lorsqu'il y'a N conditions à vérifier.

Cette structure peut se présenter de la façon suivante :

```
int Identificateur ;
if (condition1)   identificateur=1 ;
if (condition2)   identificateur=2 ;

if (conditionN-1)   identificateur=N-1 ;

if (conditionN)   identificateur=N ;

    switch(identificateur)
    {
    case 1 :{traitement1 ;break ;}
    case 2 :{traitement2 ;break ;}
    .
    .
    case N-1 :{traitementN-1 ;break ;}
    default:{traitementN ;break ;}
    }
```

B- Combinaison de conditions

B-1 Combinaison de conditions : l'opérateur logique ET (&&)

Exemple

```
If( note >=  {
  if( note < 10)  {
     printf ("vous devez passer le rattrapage du module \n");
}
```

Ces deux si imbriqués reflètent une combinaison de deux conditions qui peut se traduire par si la valeur de note est supérieure ou égale à cinq et que la valeur de note est inférieure à dix, alors effectuer l'instruction, ce que l'on écrira :

```
If (( note >= 5)) &&( note <=10))  {
   printf ("vous devez passer le rattrapage du module \n ");
}
```

B-2 Combinaison de conditions : l'opérateur logique OU (||)

Pour exprimer l'alternative, on utilise l'opérateur **ou**, la condition du si sera vraie si au moins une des sous-conditions est vraie. Il faut donc que toutes les sous-conditions soient fausses pour que le sinon soit exécuté.

Exemple

```
If( (x < 0) || (x == 0)) {
printf ("l inverse de la racine carre de votre nombre x ne peut pas être calculee \n");
}
else {
   printf ("l inverse de la racine carre de votre nombre x vaut : %f \n", 1/sqrt(x));
}
```

Il est possible de donner plusieurs conditions :

```
If((( cd1) || ( cd2)  || ( cd3) {
   traitement1 ;
}
else {
   traitement2 ;

}
```

B-3 Combinaison de conditions : l'opérateur logique NON (!( ))

L'opérateur non représente la négation, il peut s'appliquer à une variable ou à une condition à évaluer. En effet, dire "X<0" est une négation de "X>=0".

```
if (!(X<0))  {
  printf (" Votre nombre X a une  racine carre \n'');
}
else {
printf (" Votre nombre X n'a pas de  racine carre \n'');
}
```

C) Utilisation du type bool

Ce type permet de stocker le résultat d'une condition dans une variable dont le type est bool.

Cette variable vaux alors soit la valeur vrai soit la valeur faux.

Une variable de type bool se déclare de la manière suivante :

Bool nom_de_la_variable ;

D) Travail à effectuer

EX1_TP2: Ecrire, compiler et exécutez un programme qui reçoit deux réels x et y et affiche
• « Egaux » si, les deux réels sont alors considérés comme égaux.
• « Différent » si, les deux réels sont alors considérés comme différents.
en utilisant la structure alternative simple répétée puis la structure alternative complète et la combinaison de conditions

EX2_TP2:
Un émetteur de couleur délivre une couleur en fonction d'un chiffre N donné.

Couleur	Noir	Bleu	Vert	jaune	Rouge	Blanc
N	1	2	3	4	5	6

Ecrire, compiler et exécutez un programme qui donne en fonction de la valeur de N lue au clavier, la couleur correspondante. Dans le cas où N est différent des valeurs données ci-dessus, on affichera « couleur non prédéfinies ».

1) Avec la structure alternative simple répétée
2) Avec la structure alternative imbriquée
3) Avec la structure alternative de cas
4) En utilisant le type bool

Solution

```
// corrigé EX1_TP2 avec S.A.S.R
#include<stdio.h>
#include<stdlib.h>
int main()
{
//declaration
   float X,Y;
//preparation du traitement
printf("donner deux reels quelconques X et Y\n");
scanf("%f%f",&X,&Y);
//traitement
if(X==Y)
printf(" X et Y sont Egaux \n");
            if(X!=Y)
            printf("X et Y sont differents \n");
}
```

```
// corrigé EX1_TP2  avec S.A.complète
#include<stdio.h>
#include<stdlib.h>
int main()
{
//declaration
   float X,Y;
//preparation du traitement
printf("donner deux reels quelconques X et Y\n");
scanf("%f%f",&X,&Y);
//traitement
if(X==Y)
printf(" X et Y sont Egaux \n");
            else
            printf("X et Y sont differents \n");
}
```

```
// corrigé EX1_TP2 avec combinaison de condition
#include<stdio.h>
#include<stdlib.h>
int main()
{
//declaration
   float X,Y;
//preparation du traitement
printf("donner deux reels quelconques X et Y\n");
scanf("%f%f",&X,&Y);
//traitement
if((X>Y)||(X<Y))
            printf("X et Y sont differents \n");
            if(X==Y)
printf(" X et Y sont Egaux \n");
}
```

```
// Corrigé EX2_TP2_3
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
int main()
{
//declaration
   int I,N;
//preparation du traitement
printf("donner N \n");
scanf("%d",&N);
//traitement
      if(N==1)   I=1;
```

```
      if(N==2)  I=2;
      if(N==3)  I=3;
      if(N==4)  I=4;
      if(N==5)  I=5;
      if(N==6)  I=6;
switch(I)
{
 case 1:
{ printf(" Noir \n");
 break;
}
case 2:
{
printf(" Bleu \n");
break;
}
case 3:
{
printf(" Vert\n");
break;
}
case 4:
{ printf(" Jaune \n");
 break;
}
case 5:
{
printf(" Rouge \n");
break;
}
case 6:
{
printf(" Blanc\n");
break;
}
default:
{
printf(" Couleur non predefinie\n");
break;
}
}
printf("FIN \n");
}
```

```
/*corrigé EX2_TP2_4   */
#include<stdio.h>
#include<stdlib.h>
#include<stdbool.h>
int main()
{
//declaration
int N;
   bool C1,C2,C3,C4,C5,C6,C7;
//preparation du traitement
printf("donner N \n");
scanf("%d",&N);
//traitement
      C1=(N==1) ;
      C2=(N==2)  ;
      C3=(N==3)  ;
      C4=(N==4)  ;
      C5=(N==5)  ;
      C6=(N==6)  ;
      if(C1) printf(" Noir \n") ; else
      if(C2) printf(" Bleu \n") ; else
      if(C3) printf(" Vert\n") ;  else
      if(C4) printf(" Jaune \n") ;else
      if(C5) printf(" Rouge \n") ;else
      if(C6) printf(" Blanc\n") ;else
            printf(" Couleur non predefinie\n");
printf("FIN \n");}
```

Travaux pratiques numéro 3
Structures de boucle

I- but

Dans ce TP, il s'agit d'apprendre à utiliser la structure de boucle avec ses différentes variantes.

II- Quelques notions à savoir

A- Les différents cas de structure itérative

A-1 La structure itérative « for »

La syntaxe de cette structure est la suivante :

int id ;

for(id=valeur initiale ; condition qui fixe la valeur Finale de id ; id=id+pas)

{traitement à faire ;}

A-2 La structure itérative « while »

La syntaxe de cette structure est la suivante :

while (condition) **{**traitement;}

A-3 La structure itérative « do while »

La syntaxe de cette structure est la suivante :

Do traitement **while** (condition) ;

B) Travail à effectuer

EX1_TP3: Ecrire, compiler et exécutez un programme qui permet de Calculer CNP=N !/(P !)((N-P) ! en utilisant la structure **for**

EX2_TP3: Reprendre l'exercice précédent en utilisant la structure **while.** On l'utilisera aussi lors de la lecture des entiers N et P de manière à n'en tenir compte que s'ils vérifient les conditions suivantes : N>=P et N et P strictement positifs.

EX3_TP3: Reprendre l'exercice précédent en utilisant la structure **do while.** On l'utilisera aussi lors de la lecture des entiers N et P de manière à n'en tenir compte que s'ils vérifient les conditions suivantes : N>=P et N et P strictement positifs.

Solution

```
/* EX1_TP3 calcul de CNP=N!/(P!)*((N-P)!)
avec la boucle for   */
#include<stdio.h>
#include<stdlib.h>
int main()
{
//declaration
   int I,N,P,NMP;
  float FACTN,FACTP,FACTNMP,CNP;

//preparation du traitement
printf("donner un entier N>0\n");
scanf("%d",&N);
printf("donner un entier P>0 et <N\n");
scanf("%d",&P);
//traitement
FACTN=1;FACTP=1;FACTNMP=1;  //initialisation
//calcul de N!
for(I=2;I<=N;I++)
    FACTN=FACTN*I;
    printf("FACTN=%f \n", FACTN);
//calcul de P!
    for(I=2;I<=P;I++)
    FACTP=FACTP*I;
    printf("FACTP=%f \n", FACTP);
//calcul de (N-P)!
NMP=N-P;
    for(I=2;I<=NMP;I++)
    FACTNMP=FACTNMP*I;
    printf("FACTNMP=%f \n", FACTNMP);
//calcul de CNP
CNP=FACTN/((FACTP)*(FACTNMP));
printf("CNP=%f \n", CNP);
printf("FIN \n");
}

/*EX2_TP3 calcul de CNP=N!/(P!)*((N-P)!)
avec la boucle do while   */
#include<stdio.h>
#include<stdlib.h>
int main()
{
//declaration
  int I,N,P,NMP;
  float FACTN,FACTP,FACTNMP,CNP;
```

```
//preparation du traitement
do
{
printf("donner un entier N>0\n");
scanf("%d",&N);
printf("donner un entier P>0 et <N\n");
scanf("%d",&P);
}
while((N<=0)||(P<=0)||(N<P));
//traitement
FACTN=1;I=1;  //initialisation
//calcul de N!
do
{
I=I+1;
    FACTN=FACTN*I;
    }
    while(I<N);
    printf("FACTN=%f \n", FACTN);
//calcul de P!
FACTP=1;I=1;   //initialisation
do
{
    I=I+1;
    FACTP=FACTP*I;
    }
    while(I<P);
    printf("FACTP=%f \n", FACTP);
//calcul de (N-P)!
NMP=N-P;
FACTNMP=1;I=1;  //initialisation
    do
    {I=I+1;
    FACTNMP=FACTNMP*I;
    }
    while(I<NMP);
    printf("FACTNMP=%f \n", FACTNMP);
//calcul de CNP
CNP=FACTN/((FACTP)*(FACTNMP));
printf("CNP=%f \n", CNP);
printf("FIN \n");
}
```

```
/*EX3_TP3 calcul de CNP=N!/(P!)*((N-P)!)
avec la boucle while   */
#include<stdio.h>
#include<stdlib.h>
```

```
int main()
{
//declaration
   int I,N,P,NMP;
   float FACTN,FACTP,FACTNMP,CNP;

//preparation du traitement

printf("donner un entier N>0\n");
scanf("%d",&N);
printf("donner un entier P>0 et <N\n");
scanf("%d",&P);

while((N<=0)||(P<=0)||(N<P))
{printf("donner un entier N>0\n");
scanf("%d",&N);
printf("donner un entier P>0 et <N\n");
scanf("%d",&P);
}
//traitement
FACTN=1;I=1;  //initialisation
//calcul de N!
while(I<N)
{
I=I+1;
    FACTN=FACTN*I;
    }
    printf("FACTN=%f \n", FACTN);
//calcul de P!
FACTP=1;I=1;   //initialisation
while(I<P)
{
    I=I+1;
    FACTP=FACTP*I;
    }

    printf("FACTP=%f \n", FACTP);
//calcul de (N-P)!
NMP=N-P;
FACTNMP=1;I=1;  //initialisation
    while(I<NMP)
    {I=I+1;
    FACTNMP=FACTNMP*I;
    }

    printf("FACTNMP=%f \n", FACTNMP);
//calcul de CNP
CNP=FACTN/((FACTP)*(FACTNMP));
```

```
printf("CNP=%f \n", CNP);
printf("FIN \n");
}
```