

Filière d'ingénieur :

Ingénierie Informatique, Big Data et Cloud Computing (II-BDCC)

Département Mathématiques et Informatique

Compte rendu TP1

DE LA CONCEPTION ET LA PROGRAMMATION

ORIENTE OBJET EN C++

Par: BOUZINE Ahmed houssam (II-BDCC1)

Encadré par : M. K. MANSOURI

Année Universitaire : 2022/2023

Exercice 1 :

Enoncé :

Ecrire le programme suivant en C++, en ne faisant appel qu'aux nouvelles possibilités d'entrées-sorties du C++, donc en remplaçant les appels à *printf* et *scanf*

```
#include <stdio.h>

main()
{
    int n;
    float x;

    printf("donnez un entier et un flottant\n");
    scanf("%d %e", &n, &x);
    printf("le produit de %d par %e\n est %e\n", n, x, n*x);
}
```

Solution :

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main(int argc, char** argv){
6      int n;
7      float x;
8
9      cout << "donnez un entier et un flottant\n" << endl; // afficher un message a l'utilisateur pour saisir un nombre entier et un autre flottant
10
11     cin >> n >> x; // la saisie des deux nombres ( L'entier et Le flottant)
12
13     cout << "le produit de " << n << " par " << x << endl << "est : " << x*n << endl; // affichage du produit des deux nombres
14
15 }
```

Exécution :

```
C:\Users\houss\Desktop\ENSET\s2\cpp\tp\td1ex1.exe
donnez un entier et un flottant
2
3.4
le produit de 2 par 3.4
est : 6.8

-----
Process exited after 4.536 seconds with return value 0
Press any key to continue . . .
```

Exercice 2 :

Enoncé :

Tester le programme suivant :

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

void main()
{
    int i, n=25, *p;
    char *CH="On est à l'IGA !";
    float x = 25.359;

    cout<<"BONJOUR\n";
    cout<<CH<<"\n";
    cout<<"BONJOUR\n"<<CH<<"\n";
    cout<<"n= "<<n<<" x= "<<x<<" p= "<<p<<"\n";

    getch() ;
}
```

Solution :

```
1  #include <iostream>
2  #include <conio.h>
3  using namespace std;
4  main()
5  {
6      int i, n=25, *p;
7      char *CH="On est à l'IGA !";
8      float x = 25.359;
9      cout<<"BONJOUR";
10     cout<<CH<<"\n";
11     cout<<"BONJOUR\n"<<CH<<"\n";
12     cout<<"n= "<<n<<" x= "<<x<<" p= "<<p<<"\n";
13     getch() ;
14 }
```

On remarque que « cout » ne peut pas lire des caractères spécifiques comme la saute du ligne,

Après le fixage du problème on aura l'exécution suivante :

Exécution :

```
C:\Users\houss\Desktop\ENSET\s2\cpp\tp\td1ex2.exe
BONJOUROn est à l'IGA !
BONJOUR
On est à l'IGA !
n= 25 x= 25.359 p= 0
```

Exercice 3 :

Enoncé :

Tester le programme suivant plusieurs fois, en commettant des erreurs de saisie :

```
#include <iostream.h>
#include <conio.h>
void main()
{
    int n;
    char tc[30],c;
    float x;
    cout<<"Saisir un entier:";
    cin>>n;
    cout<<"Saisir un réel:";
    cin>>x;
    cout<<"Saisir une phrase:";
    cin>>tc;
    cout<<"Saisir une lettre:";
    cin>>c;
    cout<<"Affichage : "<<n<<" "<<x<<" "<<tc<<" "<<c<<"\n";
    getch() ;
}
```

Solution :

1.

```
1  #include <iostream>
2  #include <conio.h>
3
4  using namespace std;
5  void main()
6  {
7      int n; // declaration d'un nombre entier
8      char tc[30],c; // declaration d'un tableau des entiers et un tableau des caracteres et un caractere c
9      float x; // declaration d'un nombre flottant
10     cout<<"Saisir un entier:"; // message pour saisir un nombre entier
11     cin>>n; //saisie d'un nombre entier
12     cout<<"Saisir un réel:"; // message pour saisir un nombre flottant
13     cin>>x; //
14     cout<<"Saisir une phrase:"; //message pour saisir une phrase
15     cin>>tc; // copier de la phrase dans le tableau
16     cout<<"Saisir une lettre:"; // message pour saisir une lettre
17     cin>>c; // saisie du caractere
18     cout<<"Affichage : "<<n<<" "<<x<<" "<<tc<<" "<<c<<"\n"; // affichage des informations saisies
19     getch() ;
20 }
```

2.

On remarque qu'on ne peut pas copier des valeurs dans d'autres qui ont des types différents.

Exercice 4 :

Enoncé :

- 1- Ecrire une fonction puissance ayant deux paramètres d'entrée x et n et permettant de renvoyer la puissance nième de x : X^n .
- 2- Ecrire un programme principal main, permettant d'appeler cette fonction et lui passant des paramètres de types différents de ceux de sa définition..
- 3- Exécuter le programme et conclure ?
- 4- Reprendre la fonction puissance. Par défaut, la fonction puissance devra fournir x^4 .

Solution :

```
1  #include <iostream>
2  using namespace std;
3
4  float puissance(float x,int n)
5  {
6      float puissance = 1;
7      for(int i=1;i<=n;i++)
8          puissance = puissance * x;
9      return puissance;
10 }
11
12 int main()
13 {
14     int n;
15     int nbr_entier;
16     float nbr_flottant;
17     char caractere;
18     long nbr_entier_long;
19     unsigned int nbr_entier_non_signe;
20     float p1,p2,p3,p4,p5;
21
22     cout << "saisir la puissance n" << endl;
23     cin >> n ;
24     cout << "saisir un nombre entier" << endl;
25     cin >> nbr_entier;
26     p1 = puissance(nbr_entier,n);
27     cout << "saisir un nombre flottant" << endl;
28     cin >> nbr_flottant ;
29     p2 = puissance(nbr_flottant,n);
30     cout << "saisir un nombre entier long" << endl;
31     cin >> nbr_entier_long ;
32     p3 = puissance(nbr_entier_long,n);
33     cout << "saisir un nombre entier non signe" << endl;
34     cin >> nbr_entier_non_signe ;
35     p4 = puissance(nbr_entier_non_signe,n);
36     cout << "saisir un caractere" << endl;
37     cin >> caractere ;
38     p5 = puissance(caractere,n);
39     cout << "p1 = " << p1<<endl;
40     cout << "p2 = " << p2<<endl;
41     cout << "p3 = " << p3<<endl;
42     cout << "p4 = " << p4<<endl;
43     cout << "p5 = " << p5<<endl;
44 }
45
```

Exécution :

```
C:\Users\houss\Desktop\ENSET\s2\cpp\tp\td1ex4.exe
saisir la puissance n
3
saisir un nombre entier
2
saisir un nombre flottant
3.4
saisir un nombre entier long
3232
saisir un nombre entier non signe
323
saisir un caractere
f
p1 = 8
p2 = 39.304
p3 = 3.37609e+010
p4 = 3.36983e+007
p5 = 1.06121e+006
-----
Process exited after 10.32 seconds with return value 0
Press any key to continue . . .
```

3. La fonction peut marcher avec différents types de variables passant en paramètres.

4.

```
1  #include <iostream>
2  using namespace std;
3
4  float puissance(float x,int n)
5  {
6      float puissance = 1;
7      for(int i=1;i<=n;i++)
8          puissance = puissance * x;
9      return puissance;
10 }
11
12 int main()
13 {
14     int nbr_entier;
15     float nbr_flottant;
16     char caractere;
17     long nbr_entier_long;
18     unsigned int nbr_entier_non_signe;
19     float p1,p2,p3,p4,p5;
20
21     cout << "saisir un nombre entier" << endl;
22     cin >> nbr_entier;
23     p1 = puissance(nbr_entier,4);
24     cout << "saisir un nombre flottant" << endl;
25     cin >> nbr_flottant;
26     p2 = puissance(nbr_flottant,4);
27     cout << "saisir un nombre entier long" << endl;
28     cin >> nbr_entier_long;
29     p3 = puissance(nbr_entier_long,4);
30     cout << "saisir un nombre entier non signe" << endl;
31     cin >> nbr_entier_non_signe;
32     p4 = puissance(nbr_entier_non_signe,4);
33     cout << "saisir un caractere" << endl;
34     cin >> caractere;
35     p5 = puissance(caractere,4);
36     cout << "p1 = " << p1 << endl;
37     cout << "p2 = " << p2 << endl;
38     cout << "p3 = " << p3 << endl;
39     cout << "p4 = " << p4 << endl;
40     cout << "p5 = " << p5 << endl;
41 }
```

```
C:\Users\houss\Desktop\ENSET\s2\cpp\tp\td1ex4.exe
saisir un nombre entier
34
saisir un nombre flottant
3.4
saisir un nombre entier long
4343
saisir un nombre entier non signe
43
saisir un caractere
f
p1 = 1.33634e+006
p2 = 133.634
p3 = 3.55762e+014
p4 = 3.4188e+006
p5 = 1.08243e+008
-----
Process exited after 10.84 seconds with return value 0
Press any key to continue . . .
```

Exercice 5 :

Enoncé :

- 1- Tester le programme suivant :

```
#include <iostream.h>
#include <conio.h>
void test(int n = 0,float x = 2.5)
{
    cout <<"Fonction N°1 : ";
    cout << "n= "<<n<<"  x="<<x<<"\n";
}

void test(float x = 4.1,int n = 2)
{
    cout <<"Fonction N°2 : ";
    cout << "n= "<<n<<"  x="<<x<<"\n";
}
```

```
void main()
{
    int i = 5; float r = 3.2;
    test(i,r);    // fonction N°1
    test(r,i);    // fonction N°2
    test(i);      // fonction N°1
    test(r);      // fonction N°2

    // les appels suivants, ambigües, sont rejetés par le
    // compilateur
    // test();
    // test (i,i);
    // test (r,r);
    // les initialisations par défaut de x à la valeur 4.1
    // et de n à 0 sont inutilisables
    getch() ;
}
```

Solution :

```
td1ex5.cpp
1  #include <iostream>
2  #include <conio.h>
3  using namespace std;
4  void test(int n = 0, float x = 2.5)
5  {
6      cout << "Fonction N°1 : ";
7      cout << "n= " << n << " x=" << x << "\n";
8  }
9  void test(float x = 4.1, int n = 2)
10 {
11     cout << "Fonction N°2 : ";
12     cout << "n= " << n << " x=" << x << "\n";
13 }
14
15 main()
16 {
17     int i = 5; float r = 3.2;
18     test(i,r); // fonction N°1
19     test(r,i); // fonction N°2
20     test(i); // fonction N°1
21     test(r); // fonction N°2
22     // Les appels suivants, ambigus, sont rejetés par le
23     // compilateur
24     // test();
25     // test (i,i);
26     // test (r,r);
27     // Les initialisations par défaut de x à la valeur 4.1
28     // et de n à 0 sont inutilisables
29     getch() ;
30 }
```

Exécution :

C:\Users\houss\Desktop\ENSET\s2\cpp\tp\td1ex5.exe

```
15 Fonction N 1 : n= 5 x=3.2
Fonction N 2 : n= 5 x=3.2
Fonction N 1 : n= 5 x=2.5
Fonction N 2 : n= 2 x=3.2
```

2.

On peut conclure qu'on peut déclarer des fonctions avec les mêmes noms mais des types différents et lors de l'appel selon les paramètres le compilateur peut savoir à quelle fonction on fait appel.

Exercice 6 :

Enoncé :

1- Tester le programme suivant :

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

void essai(float x,char c,int n=0)
{
    cout <<"Fonction N°1: x = " << x <<" c = " << c
        <<" n = " << n << "\n";
}

void essai(float x,int n)
{
    cout <<"Fonction N°2 : x = " << x <<" n = " << n <<"\n";
}

void main()
{
    char l='z';
    int u=4;
    float y = 2.0;
    essai(y,l,u); // fonction N°1
    essai(y,l);   // fonction N°1
    essai(y,u);   // fonction N°2
    essai(u,u);   // fonction N°2
    essai(u,l);   // fonction N°1
    // essai(y,y); rejet par le compilateur
    essai(y,y,u); // fonction N°1
    getch();
}
```

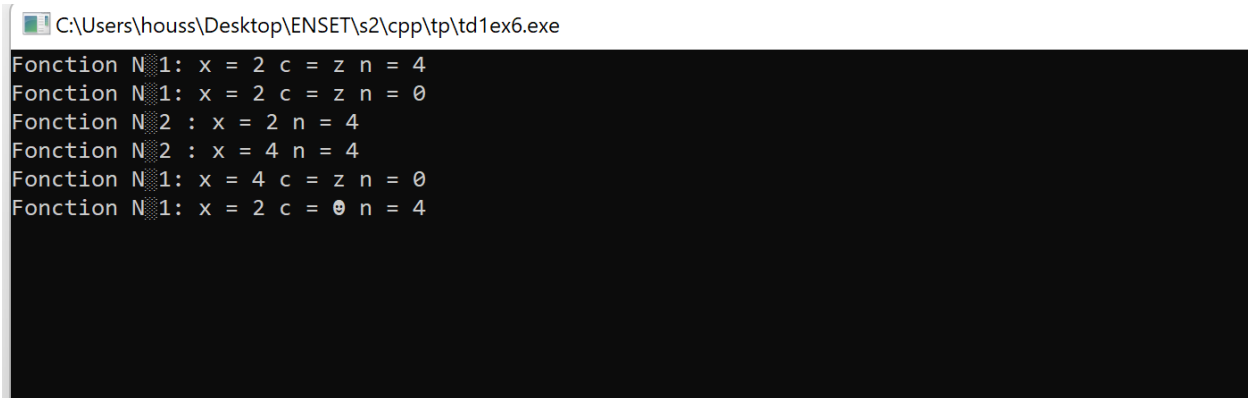
Solution :

```
#include <iostream>
#include <conio.h>
using namespace std;
void essai(float x,char c,int n=0)
{
    cout <<"Fonction N°1: x = " << x <<" c = " << c
        <<" n = " << n << "\n";
}

void essai(float x,int n)
{
    cout <<"Fonction N°2 : x = " << x <<" n = " << n <<"\n";
}

main()
{
    char l='z';
    int u=4;
    float y = 2.0;
    essai(y,l,u); // fonction N°1
    essai(y,l); // fonction N°1
    essai(y,u); // fonction N°2
    essai(u,u); // fonction N°2
    essai(u,l); // fonction N°1
    // essai(y,y); rejet par le compilateur
    essai(y,y,u); // fonction N°1
    getch();
}
```

Exécution :



```
C:\Users\houss\Desktop\ENSET\s2\cpp\tp\td1ex6.exe
Fonction N°1: x = 2 c = z n = 4
Fonction N°1: x = 2 c = z n = 0
Fonction N°2 : x = 2 n = 4
Fonction N°2 : x = 4 n = 4
Fonction N°1: x = 4 c = z n = 0
Fonction N°1: x = 2 c = 0 n = 4
```

On peut conclure que C++ contient une nouvelle notion à propos des fonctions cette notion est la surcharge.

Exercice 7 :

Enoncé :

- 1- Ecrire une fonction : void affiche (float x, int n = 0) qui affiche x^n .
(Avec en particulier $x^0 = 1$ et donc, $0^0 = 1$),
- 2- Ecrire une autre fonction : void affiche(int n, float x=0) qui affiche x^n .
(Avec en particulier $0^n = 0$ et donc, $0^0 = 0$).
- 3- Appeler ces fonctions dans un programme principal, en utilisant la propriété de surdéfinition.

Solution :

```

1  #include <iostream>
2  #include <conio.h>
3  using namespace std;
4
5  void affiche(float x, int n = 0)
6  {
7      if (n == 0)
8      {
9          cout << " resultat = 1" << endl;
10     }
11
12     float puissance = 1;
13     for (int i = 1; i <= n; i++)
14     {
15         puissance = puissance * x;
16     }
17     cout << "Le resultat  puissance = " << puissance << endl;
18 }
19
20 void affiche(int n, float x = 0)
21 {
22     if (x == 0)
23     {
24         cout << "Le resultat = 0" << endl;
25     }
26     else{
27         float puissance = 1;
28         for (int i = 1; i <= n; i++)
29         {
30             puissance = puissance * x;
31         }
32         cout << "Le resultat  puissance = " << puissance << endl;
33     }
34 }
35
36
37 main()
38 {
39     int n, n1 = 0;
40     float base, m = 0;
41     cout << "saisir la base x comme nombre flottant" << endl;
42     cin >> base;
43     cout << "saisir la puissance n" << endl;
44     cin >> n;
45     cout << "le resultat de cet appel affiche(x) avec x est un nombre flottant est : " << endl;
46     affiche(base);
47     cout << "le resultat de cet appel affiche(x) avec x est un nombre entier est : " << endl;
48     affiche(n) ;
49     cout << "le resultat de cet appel affiche(x,n) est : " << endl;
50     affiche(base,n);
51     cout << "le resultat de cet appel affiche(n,x) est : " << endl;
52     affiche(n,base);
53     getch();
54 }

```

Exercice 8 :

Solution :

```
1  #include <iostream>
2  #include <conio.h>
3  using namespace std;
4  void exchange(int a,int b)
5  {
6      int tampon;
7      tampon = b; b = a; a = tampon;
8      cout<<"Pendant l'échange: a = "<<a<<" b = "<<b<<"\n";
9  }
10
11  main()
12  {
13      int u=5,v=3;
14      cout<<"Avant échange: u = "<<u<<" v = "<<v<<"\n";
15      exchange(u,v);
16      cout<<"Après échange: u = "<<u<<" v = "<<v<<"\n";
17      getch();
18  }
```

C:\Users\houss\Desktop\ENSET\s2\cpp\tp\td1ex8.exe

```
Avant échange: u = 5 v = 3
Pendant l'échange: a = 3 b = 5
Après échange: u = 5 v = 3
```

On remarque que les variables gardent leurs valeurs principales.

```
#include <iostream>
#include <conio.h>
using namespace std;
void exchange(int *a,int *b)
{
    int tampon;
    tampon = *b; *b = *a; *a = tampon;
    cout<<"Pendant l'échange: a = "<<*a<<" b = "<<*b<<"\n";
}
main()
{
    int u=5,v=3;
    cout<<"Avant échange: u = "<<u<<" v = "<<v<<"\n";
    exchange(&u,&v);
    cout<<"Après échange: u = "<<u<<" v = "<<v<<"\n";
    getch();
}
```

C:\Users\houss\Desktop\ENSET\s2\cpp\tp\td1ex8.exe

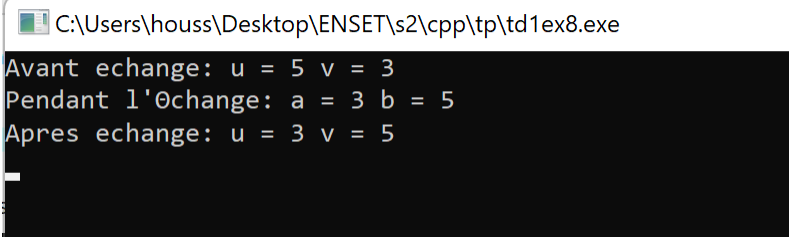
```
Avant échange: u = 5 v = 3
Pendant l'échange: a = 3 b = 5
Après échange: u = 3 v = 5
```

On remarque que chaque variable prend la valeur de l'autre variable.

```

#include <iostream>
#include <conio.h>
using namespace std;
void exchange(int &a,int &b)
{
    int tampon;
    tampon = b; b = a; a = tampon;
    cout<<"Pendant l'échange: a = "<<a<<" b = "<<b<<"\n";
}
main()
{
    int u=5,v=3;
    cout<<"Avant échange: u = "<<u<<" v = "<<v<<"\n";
    exchange(u,v);
    cout<<"Après échange: u = "<<u<<" v = "<<v<<"\n";
    getch() ;
}

```



```

C:\Users\houss\Desktop\ENSET\s2\cpp\tp\td1ex8.exe
Avant échange: u = 5 v = 3
Pendant l'0change: a = 3 b = 5
Après échange: u = 3 v = 5

```

On remarque que chaque variable prend la valeur de l'autre variable.

Exercice 9 :

Enoncé :

Soit le modèle de structure suivant :

```

struct essai
{
    int n;
    float x;
};

```

Ecrire une fonction nommée `Remise_a_zero`, permettant de remettre à zéro les 2 champs d'une structure de ce type, transmise en argument :

- par adresse
- par référence

Dans les deux cas, on écrira un petit programme d'essai de la fonction, il affichera les valeurs d'une structure de ce type, après appel de la dite fonction.

Solution :

```
1  #include <iostream>
2  #include <conio.h>
3  #include <malloc.h>
4  using namespace std;
5
6  struct essai
7  {
8      int n;
9      float x;
10 };
11
12 void Remise_a_zero(struct essai *essai){
13     essai->n = 0 ;
14     essai->x = 0.0;
15 }
16
17 void Remise_a_zero1(struct essai &essai){
18     essai.n = 0 ;
19     essai.x = 0.0;
20 }
21
22 main()
23 {
24     struct essai *ess;
25     ess= (essai*)malloc(sizeof(essai));
26     struct essai essai1;
27     Remise_a_zero(ess);
28     Remise_a_zero1(essai1);
29     cout << "valeurs apres remise a zero en utilisant le passage par adresse sont : " << ess->n << " " << ess->x << endl;
30     cout << "valeurs apres remise a zero en utilisant le passage par reference sont : " << essai1.n << " " << essai1.x << endl;
31
32 }
```

Exécution :

```
C:\Users\houuss\Desktop\ENSET\s2\cpp\tp\td1ex9.exe
valeurs apres remise a zero en utilisant le passage par adresse sont : 0 0
valeurs apres remise a zero en utilisant le passage par reference sont : 0 0
-----
```