Université Hassan II de Casablanca
ENSET de Mohammedia
Département Mathématiques et Informatique
Filière d'Ingénieurs : GLSID 1 & BDCC 1
Prof. M. K. MANSOURI

TP 1
PROGRAMMATION ORIENTEE OBJET EN C++
C++ COMME UN LANGAGE C AVANCE : PRINCIPAUX APPORTS

Exercice 1 :

Ecrire le programme suivant en C++, en ne faisant appel qu'aux nouvelles possibilités d'entrées-sorties du C++, donc en remplacent 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);
}
```

Exercice 2 :

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() ;
}
```

Exercice 3 :

1- 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() ;
}
```

2- conclure ?

Exercice 4:

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.

Exercice 5:

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, ambiguës, 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() ;
}
```

2- Conclure ?

Exercice 6:

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() ;
}
```

2- Conclure ?

Exercice 7:

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.

Exercice 8:

1- Tester les programmes suivants séparément :

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

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

void echange(int *a,int *b)
{
    int tampon;
    tampon = *b; *b = *a; *a = tampon;
    cout<<"Pendant l'échange: a = "<<*a<<" b = "<<*b<<"\n";
}
void main()
{
    int u=5,v=3;
    cout<<"Avant echange: u = "<<u<<" v = "<<v<<"\n";
    echange(&u,&v);
    cout<<"Apres echange: u = "<<u<<" v = "<<v<<"\n";
    getch() ;
}
```

```
#include <iostream.h>
#include <conio.h>
void echange(int &a,int &b)
{
      int tampon;
      tampon = b; b = a; a = tampon;
      cout<<"Pendant l'échange: a = "<<a<<" b = "<<b<<"\n";
}

void main()
{
      int u=5,v=3;
      cout<<"Avant echange: u = "<<u<<" v = "<<v<<"\n";
      echange(u,v);
      cout<<"Apres echange: u = "<<u<<" v = "<<v<<"\n";
      getch() ;
}
```

2- Q'est ce que vous constatez dans chaque cas ?
3- Conclure ?

Exercice 9 :

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.

Solutions du TP N° 1:

Exercice 1 :

```
#include <iostream.h>
int main(void)
{
  int n;
  float x;
  cout << "donnez un entier et un flottant\n";
  cin >> n >> x;
  cout <<"le produit de " << n << " par " << x << "\n est " <<
n*x;
  return 0;
}
```

Exercice 4 :

```
#include <iostream.h>
#include <conio.h>
float puissance(float x,int n)
{
   float R = 1;
   for(int i=1;i<=n;i++)
   R = R * x;
   return R;
}

void main()
{
   char c=5;
   int i=10,j=6;
   float r=5.246,r1,r2,r3,r4,r5;
   r1 = puissance(r,j);
   r2 = puissance(r,c);
   r3 = puissance(j,i);
   r4 = puissance(j,r);
   r5 = puissance(0,4);
   cout << "r1 = " <<r1<<"\n";
   cout << "r2 = " <<r2<<"\n";
   cout << "r3 = " <<r3<<"\n";
   cout << "r4 = " <<r4<<"\n";
   cout << "r5 = " <<r5<<"\n";
   getch() ;
}
```

Exercice 7 :

```
#include <iostream.h>
#include <conio.h>
void affiche(float x,int n=0)
{
   int i = 1;float resultat = 1;
   for(;i<=n;i++)resultat = resultat * x;


   cout << "x = "<<x<< " n = " << n << " resultat = " << resultat
<<"\n";
}

void affiche(int n,float x=0)
{
   int i = 1;float resultat = 1;
   if (n!=0){for(;i<=n;i++)resultat = resultat * x;}
   else (resultat = 0);
   cout << "n = "<<n<< " x = " << x << " resultat = " << resultat
<<"\n";
}
void main()
{
   int u=4,v=0;float y = 2.0,z=0;
   affiche(u);
   affiche(y);
   affiche(y,u);
   affiche(u,y);
   affiche(v,z);
   affiche(z,v);
   getch() ;
}
```

Exercice 7 :

Avec une transmission par adresse :

```
#include <iostream.h>
struct essai
{
  int n;
  float x;
};
void Remise_a_zero(struct essai *ads)
{
  ads->n = 0;
  ads->x = 0.0;
}
int main(void)
{
  struct essai s;
  Remise_a_zero(&s);
  cout << "valeurs après remise à zéro : " << s.n << " " << s.x
<< endl;
  return 0;
}
```

Avec une transmission par référence :

```
#include <iostream.h>
struct essai
{
  int n;
  float x;
};
void Remise_a_zero(struct essai &ads)
{
  ads.n = 0;
  ads.x = 0.0;
}
int main(void)
{
  struct essai s;
  Remise_a_zero(s);
  cout << "valeurs après remise à zéro : " << s.n << " " << s.x
<< endl;
  return 0;
}
```