

## T.P. 3

# Initiation au langage C++

## La surcharge

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

int somme (int a,int b)
{      return (a+b);  }

int somme (float a,float b)
{      return (a+b);  }

void main()
{      int n1=4;
        int n2=5;
        float r1=6.71;
        float r2=4.95;

        printf ("la somme des nombres est : %d \n",somme (n1,n2));
        printf ("la somme des nombres est : %d \n",somme (r1,r2));
        getch () ;
}
```

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

void printf (int a)
{      printf ("%d",a);      }

void printf (float a)
{      printf ("%f",a);      }

void main()
{
    float x=6.32;
    printf (5);
    printf ("\n");
    printf (x);
    getch ();
}
```

# Struct

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

struct robot                                //class robot
{
    int Xpos;
    int Ypos;
    char Orientation;
// public :
    int Xposition (void)
    {
        return (Xpos);
    }

    int Yposition (void)
    {
        return (Ypos);
    }

    void avance(int n)
    {
        switch (Orientation)
        {
            case 'N': {Ypos=Ypos+n; break;}
            case 'E': {Xpos=Xpos+n; break;}
            case 'S': {Ypos=Ypos-n; break;}
            case 'O': {Xpos=Xpos-n; break;}
        }
    }

    void tourne(void)
    {
        switch (Orientation)
        {
            case 'N': {Orientation='E'; break;}
            case 'E': {Orientation='S'; break;}
            case 'S': {Orientation='O'; break;}
            case 'O': {Orientation='N'; break;}
        }
    }

    robot ()
    {
        Xpos=0;
        Ypos=0;
        Orientation='N';
    }
};

void main (void)
{
    robot r1;
    robot r2;
    r1.avance (5); r1.tourne ();   r1.avance (2);
    r2.tourne ();   r2.avance (2); r2.avance (3);
    printf ("Robot 1 : X = %d Y = %d \n",r1.Xposition (),r1.Yposition ());
    printf ("Robot 2 : X = %d Y = %d \n",r2.Xposition (),r2.Yposition ());
}
```

# Les méthodes

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

```
class robot
{
    int Xpos;
    int Ypos;
    char Orientation;
public :
    int Xposition (void);
    int Yposition (void);
    void avance(int n);
    void tourne(void);
    robot ();
};
```

```
int robot::Xposition (void)
{
    return (Xpos); }
```

```
int robot::Yposition (void)
{
    return (Ypos); }
```

```
void robot::avance(int n)
{
    switch (Orientation) {
        case 'N': {Ypos=Ypos+n; break;}
        case 'E': {Xpos=Xpos+n; break;}
        case 'S': {Ypos=Ypos-n; break;}
        case 'O': {Xpos=Xpos-n; break;}
    }
}
```

```
void robot::tourne(void)
{
    switch (Orientation) {
        case 'N': {Orientation='E'; break;}
        case 'E': {Orientation='S'; break;}
        case 'S': {Orientation='O'; break;}
        case 'O': {Orientation='N'; break;}
    }
}
```

```
robot::robot ()
{
    Xpos=0;    Ypos=0;
    Orientation='N';
}
```

```
void main (void)
{
    robot r1;
    robot r2;
    r1.avance (5); r1.tourne ();  r1.avance (2);
    r2.tourne ();  r2.avance (2); r2.avance (3);
    printf ("Robot 1 : X = %d Y = %d \n",r1.Xposition (),r1.Yposition ());
    printf ("Robot 2 : X = %d Y = %d \n",r2.Xposition (),r2.Yposition ());
}
```

# La surcharge opérateur

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

class matrice
{public :
    int M[2][2];
    matrice(int a,int b,int c,int d);
    matrice();
    matrice operator= (matrice a);
    friend matrice operator+ (matrice a,matrice b);    };

matrice operator+ (matrice a,matrice b)
{    matrice tmp;
    tmp.M[0][0] = a.M[0][0] + b.M[0][0];
    tmp.M[0][1] = a.M[0][1] + b.M[0][1];
    tmp.M[1][0] = a.M[1][0] + b.M[1][0];
    tmp.M[1][1] = a.M[1][1] + b.M[1][1];
    return tmp;    }

matrice matrice::operator= (matrice a)
{    this->M[0][0]=a.M[0][0];
    this->M[0][1]=a.M[0][1];
    this->M[1][0]=a.M[1][0];
    this->M[1][1]=a.M[1][1];
    return *this;    }

matrice::matrice (int a,int b,int c,int d)
{    M[0][0]=a;
    M[0][1]=b;
    M[1][0]=c;
    M[1][1]=d;    }

matrice::matrice ()
{    }

void main (void)
{
    matrice M1 (1,2,3,4);
    matrice M2 (5,6,7,8);
    matrice M3;

    M3 = M1 + M2;
    printf ("M1 + M2 -> M1.1=%d M1.2=%d M2.1=%d
M2.2=%d\n",M3.M[0][0],M3.M[0][1],M3.M[1][0],M3.M[1][1]);
    getch ();
}
```