

---

**T.P. 4**

Initiation au langage C++

Initiation à Visual C++

---

DEA SyAM 2000/2001

---

Par  
Philippe Lucidarme : [lucidarm@lirmm.fr](mailto:lucidarm@lirmm.fr)

# Initiation au langage C++

## Les librairies

Vous avez déjà utilisé de nombreuses librairies tels que conio.h, math.h, etc ... Mais il vous est possible de créer votre propre librairie : saisissez le code suivant, et enregistrez le sous le nom robot.h

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

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

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

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

void robot::setorientation (char c)
    {
        Orientation=c;
    }

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';
}
```

Vous remarquerez que ce code ne comporte pas de programme principal main(). Nous venons de créer un librairie robot.h. Et toutes les classes et fonctions de cette librairie seront accessible depuis un autre programme à condition d'y inclure cette bibliothèque :

```
#include <robot.h>
```

Attention à bien préciser le chemin exact, par exemple :

```
#include <p:\DEA\mon_nom\robot.h>
```

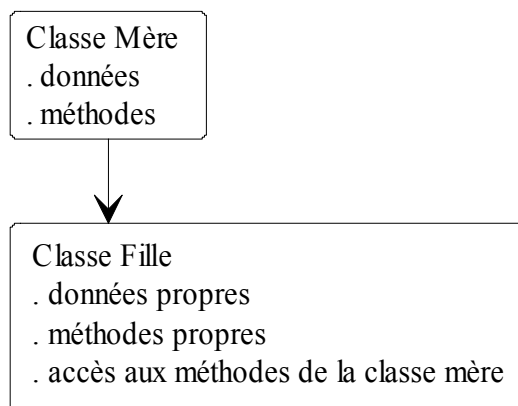
Créez un nouveau programme incluant la bibliothèque robot.h. Et faisant évoluer simultanément deux robots avant d'afficher leur positions.

## L'héritage

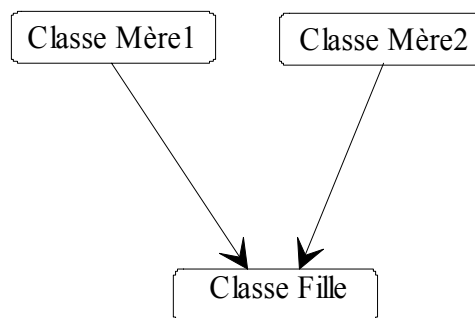
La P.O.O. ( Programmation Orientée Objet ) permet de définir de nouvelles classes (classes filles) dérivées de classes de base (classes mères), avec de nouvelles potentialités. Ceci permettra à l'utilisateur, à partir d'une bibliothèque de classes donnée, de développer ses propres classes munies de fonctionnalités propres à l'application.

On dit qu'une classe fille DERIVE d'une ou de plusieurs classes mères.

Héritage simple:



Héritage multiple:



**La classe fille n'a pas accès aux données (privées) de la classe mère.**

Saisissez le programme suivant :

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

```
class vecteur
{
    float x,y;
```

```

public: void initialise(float,float);
        void homotethie(float);
        void affiche();
};

void vecteur::initialise(float abs=0.,float ord = 0.)
{x=abs;y=ord;}

void vecteur::homotethie(float val)
{x = x*val; y = y*val;}

void vecteur::affiche()
{printf ("\n x = %2.2f  y= %2.2f ",x,y); }

class vecteur3:public vecteur
{
    float z;
public: void initialise3(float,float,float);
        void homotethie3(float);
        void hauteur(float ha){z = ha;}
        void affiche3();
} ;

void vecteur3::initialise3(float abs=0.,float ord=0.,float haut=0.)
{    initialise(abs,ord); z = haut;
}

void vecteur3::homotethie3(float val)
{    homotethie(val); z = z*val;
}

void vecteur3::affiche3()
{    affiche(); printf ("z = %2.2f",z);
}

void main()
{
    vecteur3 v, w;
    v.initialise3(5, 4, 3);  v.affiche3();
    w.initialise(8,2);      w.hauteur(7);      w.affiche();
    printf ("\n\n*****\n");

    w.affiche3(); w.homotethie3(6);  w.affiche3();
    getch() ;
}

```

Exécutez et commentez le.

Ce programme contient la classe mère Vecteur et la classe héritante vecteur3.

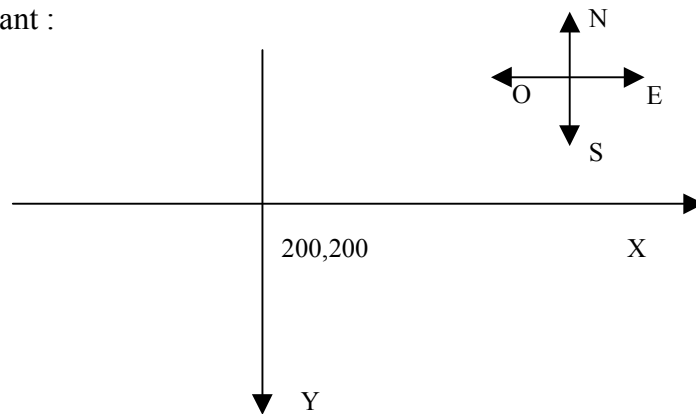
Syntaxe :

```
class nom_classe_fille:public nom_classe_mère
{
}
}
```

Syntaxe d'une fonction membre :

```
Type nom_classe_fille: :nom_fonction (paramètres)
{
}
}
```

Créez une classe rhp ( pour Robot Holonomme Peintre) dans la bibliothèque robot.h qui hérite de la classe robot. Ce robot peut se déplacer dans toutes les directions (N,E,S et O) d'un seul pas. Il peut, en plus, peindre. Il possède un pinceau qu'il peut lever ou baisser. Il se trouve dans le repère suivant :



Cette classe possède

- Une variable membre pinceau qui vaut 1 quand le pinceau est baissé et 0 quand le pinceau est levé.
- Une fonction membre av (char c) qui avance de d'un pas dans la direction qui sera passée en paramètre.
- Une fonction crayon qui permet de changer l'état du pinceau.
- Une fonction crayonstate qui renvoie l'état du pinceau.
- Une fonction Xrph qui renvoie la position en x translatée de 200 du côté positif
- Une fonction Yrph qui renvoie la position en y inversée et translatée de 200.
- Un constructeur rhp qui initialise le robot en 0,0 avec le crayon baissé.

Le programme suivant devrait permettre de tester la librairie :

```
#include <d:\sinclair\sources\cpp\robot.h>
#include <stdio.h>

void main()
{
    rhp r1;          rhp r2;
    r1.av ('N');      r1.av ('E');    r1.av ('E');    r1.crayon ();
    r2.av ('N');      r2.av ('S');    r2.crayon ();    r2.crayon ();
    printf ("Robot 1 : X = %d Y = %d ",r1.Xrhp (),r1.Yrhp ());
    if (r1.crayonstate()==1) printf ("le crayon est baissé\n");
        else
    printf ("le crayon est leve\n");
    printf ("Robot 2 : X = %d Y = %d ",r2.Xrhp (),r2.Yrhp ());
    if (r2.crayonstate()==1) printf ("le crayon est baisse\n");
        else
    printf ("le crayon est leve\n");
}
```

Vous devez obtenir le résultat suivant à l'écran si vous n'avez pas fait d'erreur :

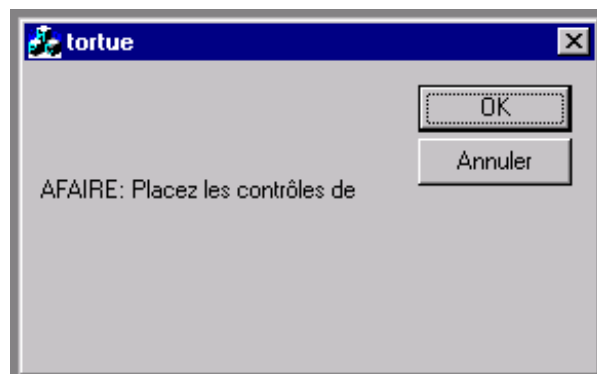
```
Robot 1 : X = 202 Y = 199 le crayon est leve
Robot 2 : X = 200 Y = 200 le crayon est baisse
```

## Initiation à Visual C++

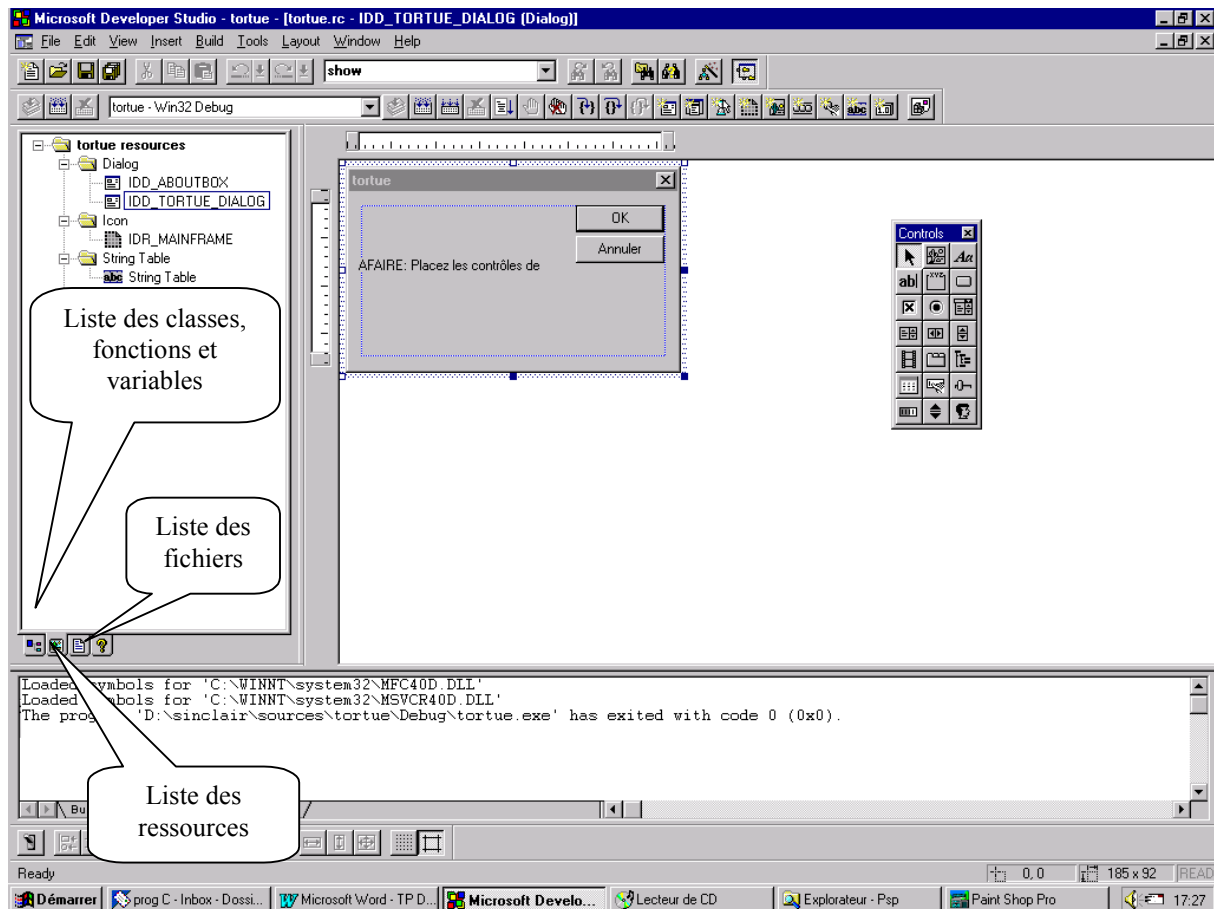
Nous allons maintenant nous intéresser à la programmation Windows. Jusqu'à présent, nos programmes étaient exécutés dans une fenêtre DOS. Nous allons voir qu'il est possible d'utiliser l'environnement Windows (plus précisément NT) pour réaliser nos programmes.

- Fermer l'espace de travail ( [File] – [Close Workspace] ) et toutes les fenêtres que vous avez ouvertes.
- Ouvrez un nouveau projet ( [File] – [New] ) et choisissez Project Workspace.
- Choisissez [MFC AppWizard] et nommez le Tortue.
- Choisissez [Dialog Based] et [Finish]
- Exécutez le programme (F5)

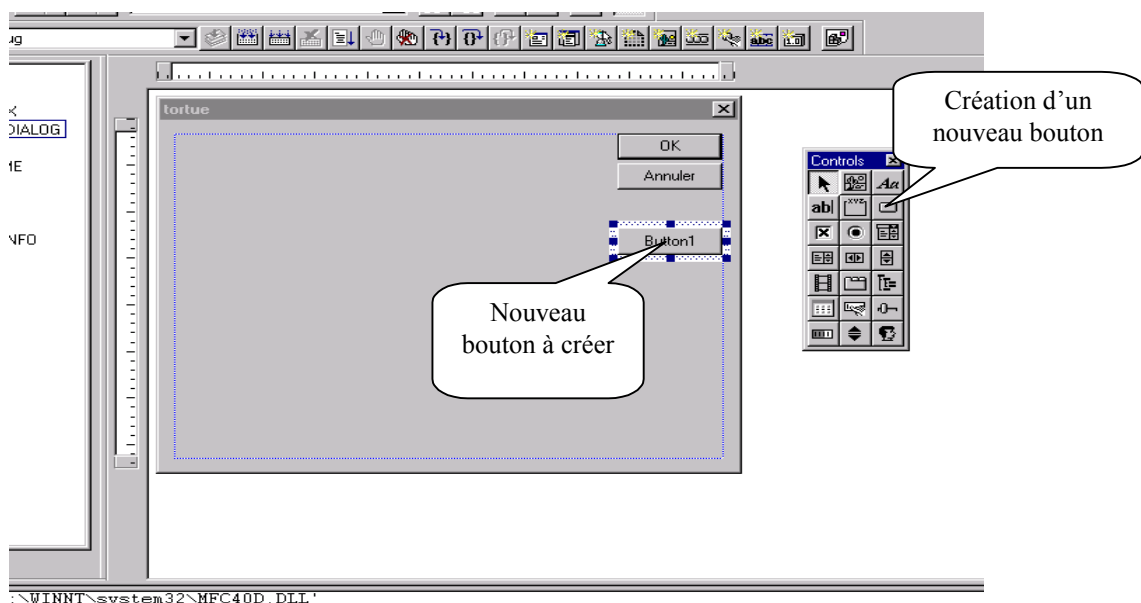
Vous venez de créer votre premier programme sous Windows, et la fenêtre suivante apparaît :



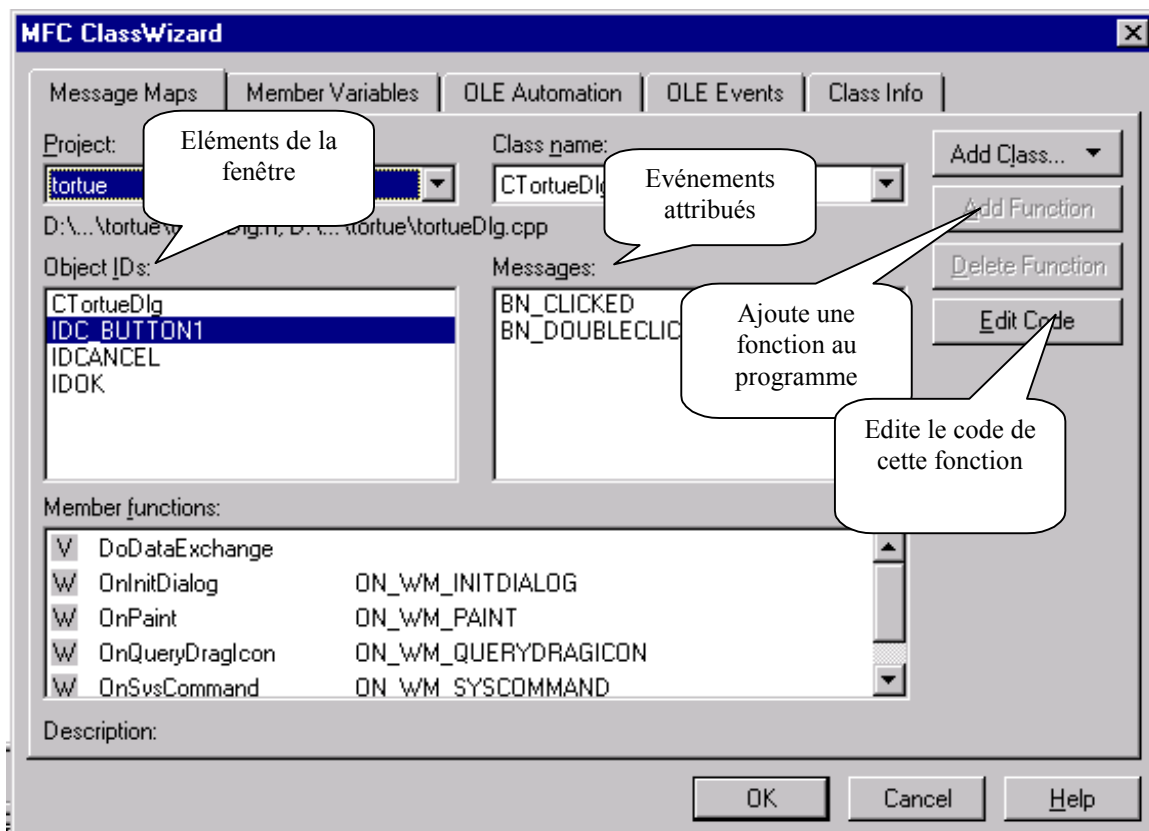
Vous venez en réalité d'utiliser un créateur automatique de programme. Celui-ci a créé un certain nombre de fichiers et de classes automatiquement. Voici une brève description de l'environnement :



- Dans les ressources, ouvrez IDD\_TORTUE\_DIALOG
- Supprimez le texte « AFAIRE: Placez les contrôles de boîtes de dialogue ici. » et agrandissez la fenêtre tortue en la sélectionnant et en tirant sur un coin.



- Maintenant, ajoutez un nouveau bouton et placez le comme sur le dessin précédent. Renommer le Dessine (Double-cliquez pour avoir ses propriétés).
- Sélectionnez le bouton, et tapez CTRL-W. Une nouvelle fenêtre apparaît :



L'élément sélectionné par défaut est le bouton Dessine [IDC\_BUTTON1]. Il peut survenir deux événements : un simple clique [BN\_CLICKED] ou un double clique [BN\_DOUBLECLICKED].

- Créez une nouvelle fonction associée au bouton Dessine, qui surviendra sur un simple clique. Puis éditez son code.
- L'éditeur ouvre automatique le fichier tortueDlg.cpp, et il a ajouté automatiquement la fonction suivante :

```
void CTortueDlg::OnButton1()
{
    // TODO: Add your control notification handler code here
}
```

- A la place du commentaire « // TODO: Add your control notification handler code here » saisissez le code suivant :

```
void CTortueDlg::OnButton1()
{
    CClientDC dc(this);
    int i, j;
    for(i=0; i<255; i++)
    {
        for(j=0; j<255; j++)
        {
            dc.SetPixel(i, j, RGB(i, j, i*j));
        }
    }
}
```

- Exécuter le programme et dans la fenêtre tortue, cliquez sur Dessine : Ô c'est zolie !

Mais revenons sur le code que nous venons de saisir :

Celui-ci va s'exécuter dès que l'utilisateur cliquera sur le bouton Dessine. L'instruction SetPixel est une fonction membre de la classe CClientDC. This représente l'élément courant (ici la fenêtre). Cette routine est donc composée de deux boucles imbriquées l'une dans l'autre et permettent d'afficher un carré de pixels de couleurs différentes. i représente l'axe des X, j l'axe des Y et RGB, les trois composantes de base de la couleur. Ici,

- la composante Rouge est i.
- la composante Verte est j.
- la composante Bleu est i x j.

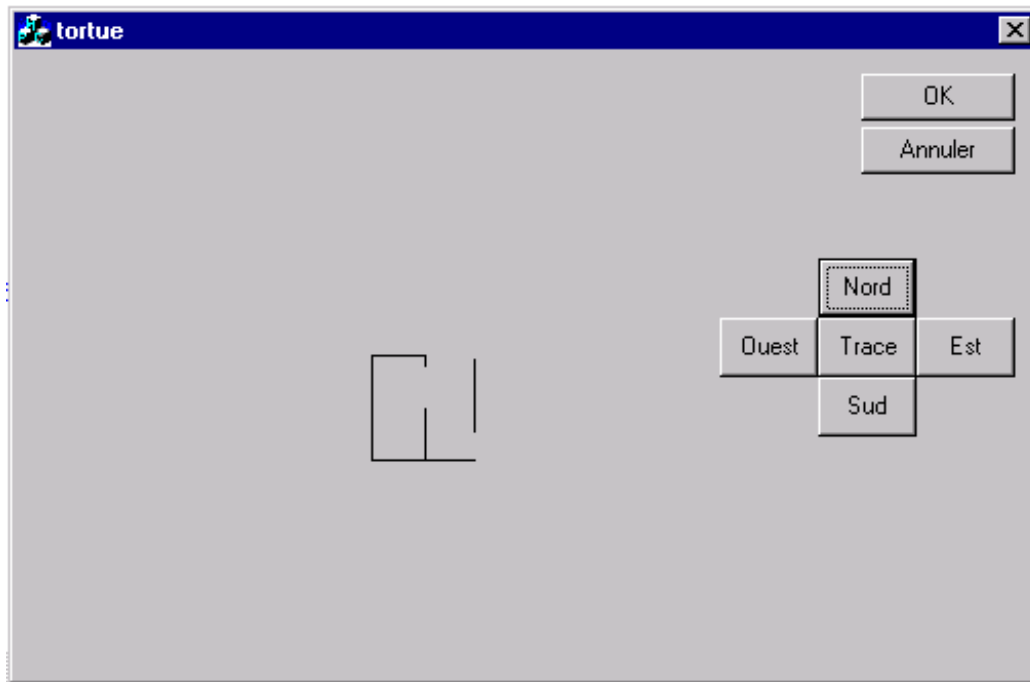
Modifiez les paramètres de cette routine afin de vous familiariser avec.

## La fameuse tortue

Pour l'instant, notre programme n'a rien à voir avec la tortue. En réalité, la tortue est un Robot Holonome Peintre. Elle peut se déplacer, mais très lentement. Elle peut à n'importe quel instant changer de direction. Et quand elle se déplace, elle laisse des traces sur son chemin, mais pas toujours !

Créez l'interface de la page suivante. A chaque bouton, associez une fonction :

- Nord : déplace la tortue vers le nord, si le crayon est baissé, la tortue dessine
- Est : déplace la tortue vers l'est, si le crayon est baissé, la tortue dessine
- Sud : déplace la tortue vers le sud, si le crayon est baissé, la tortue dessine
- Ouest : déplace la tortue vers l'ouest, si le crayon est baissé, la tortue dessine
- Trace : change l'état du crayon



Testez le programme.

Pour intégrer vos bibliothèque dans le projet, vous devez séparer la définition de la classe de ses fonctions membres. Par exemple pour la classe robot : définissez la classe dans le fichier robot.h, et ses fonctions membres dans le fichier robot.cpp. Lors de la compilation du fichier robot.cpp, celui-ci sera automatiquement ajouté à l'espace de travail.

Par exemple dans robot.h :

```
#ifndef _ROBOT_H
#define _ROBOT_H

class rob
{
    int Xpos;
    int Ypos;
    char Orientation;

public :
    rob ();
    int Xposition (void);
    int Yposition (void);
    void setorientation (char);
    void avance(int n);
    void tourne(void);
};
#endif
```

Dans robot.cpp

```
#include <stdafx.h>
#include "d:\sinclair\sources\cpp\robot.h"

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

int rob::Xposition (void)
    {    return (Xpos); }

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

void rob::setorientation (char c)
    {    Orientation=c; }

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