C++基礎　2

2006/05/09

オブジェクト指向

① オブジェクト指向とは何か？

- ソフトウェアを楽に作るための技術
- 必要な機能を探しやすくするための技術
- プログラマが楽をするための技術
- キーワードは、『クラス』『他態性』『継承』

② C＋＋で拡張された、オブジェクト指向用の機能とは何か？

- C++でオブジェクトに相当するのは『クラス』

③ クラスとは何か？

- 平たく言えば、構造体に関数をつけたもの

● C 言語で開発をしていくと、大きいプログラムになるにつれ、サブルーチンである関数の数もどんどん増えていきます。現在の Win32 と呼ばれるウィンドウズの API 群などは、C 言語で書かれていますが、既に 1 万 5 千もの関数があると言われています。そんな膨大な関数をリストアップして管理することは、非常に難しくなってきました。

● そこで、データとそれを操作する関数を一まとめにしてしまうという方法が考えられました。

例：三角形の面積を出すとき・・

C 言語

```
struct StTriangle{
    double dTeihen;
    double dTakasa;
};

double GetTriArea(const StTriangle* pstTri)
{
    return (pstTri->dTeihen * pstTri->dTakasa) * 0.5;
}

void main()
{
    struct StTriangle stTri;
    stTri.dTeihen = 10.0;
    stTri.dTakasa = 5.5;

    double dArea = GetTriArea(&stTri);
}
```

三角形用データ

三角形用関数

C++

```
class CTriangle{
public:
    CTriangle()
    :   m_dTeihen(0.0)
    ,   m_dTakasa(0.0)
    {}
    CTriangle(double dTeihen, double dTakasa)
    :   m_dTeihen(dTeihen)
    ,   m_dTakasa(dTakasa)
    {}
    void SetTeihen(double dTeihen)   { m_dTeihen = dTeihen; }
    void SetTakasa(double dTakasa)   { m_dTakasa = dTakasa; }
    double GetArea()                 { return (m_dTeihen * m_dTakasa) * 0.5; }

private:
    double m_dTeihen;
    double m_dTakasa;
};

void main()
{
    CTriangle cTri(10.0, 5.5);

    double dArea = cTri.GetArea();
}
```

三角形そのものとして扱う

④ クラスの方が書かなければならん項目が増えていないか？

- 増えます。しかしそれは局地的に見た場合です。何千、何万というコードを書く場合を考えると、クラスを使った方が書くコードは少なくなるでしょう。その理由は順次・・

⑤ 結局上記コードで何がよくなっているのか？

・ 上記コードは非常に内容が少ないため、違いが分かりません。しかし、たとえば次の中から関数を一つ探すとなると、果たしてやる気が沸くでしょうか？

（実在のライブラリヘッダより）

```
int  geo_2pt_w_to_box(double p1[XY], double p2[XY], double width, double box[5]
int  geo_2pt_w_to_rect(REAL p1x,  REAL p1y,  REAL p2x,  REAL p2y,  REAL width,
int geo_3out_prod(P3D a, P3D b, P3D c) ;
int  geo_3pt_to_rect(REAL p1x,  REAL p1y,  REAL p2x,  REAL p2y, REAL p3x, REAL
void  geo_paramat(REAL dx, REAL dy, REAL dz, MAT44 mat) ;
void  geo_rollmat(int flg, REAL ang, MAT44 mat) ;
void  geo_rollmat2(REAL  jiku[2][XYZ], REAL ang, MAT44 mat) ;
void  geo_para(REAL bp[XYZ], REAL dlt[XYZ], REAL ap[XYZ]) ;
void  geo_para2(REAL bp[XYZ], MAT44 mat, REAL ap[XYZ]) ;
void  geo_roll(REAL ln[2][XYZ], REAL bp[XYZ], REAL ang, REAL ap[XYZ]) ;
void  geo_roll2(REAL ln[2][XYZ], REAL bp[XYZ], MAT44 mat, REAL ap[XYZ]) ;
REAL  geo_ang_norm(REAL rad) ;
int geo_ang_onarc(REAL sa, REAL ea, REAL ta) ;
//2002/04/01 wata 修正
//MFCのdefineとかぶる為、引数変更 rad1→radian1
//int      geo_cir_p_cir( REAL cir1x , REAL cir1y , REAL rad1 ,
int      geo_cir_p_cir( REAL cir1x , REAL cir1y , REAL radian1 ,
//2002/04/01 wata end
REAL geo_dist(REAL xa,REAL ya,REAL xb,REAL yb) ;
long      geo_equdm2( REAL cwrk[3] ,int *ncnt ,REAL rwrk[2] ) ;
int      geo_equr21( REAL cof1[6] ,REAL cof2[3] ,int *ncnt,REAL crspnt[][2]) ;
int  geo_flip1(int n, REAL p[][XY], REAL sx, REAL sy, REAL ex, REAL ey) ;
int  geo_flip2(int n, REAL pin[][XY], REAL sx, REAL sy, REAL ex, REAL ey, REAL
REAL  geo_flip_ang(REAL ang, REAL sx, REAL sy, REAL ex, REAL ey) ;
REAL geo_in_prod(REAL vx1, REAL vy1, REAL vx2, REAL vy2) ;
int geo_inout_pnt(double pnt[XY], int n, double pl[][XY]) ;
int  geo_inverse_loop(int n, double pl[][XY]) ;
int  geo_iscross_bbox(double bbox1[MINMAX][XY], double bbox2[MINMAX][XY]) ;
int  geo_line_to_prow(int nin, REAL l[][STARTEND*XY], int *nout, REAL p[][XY])
int  geo_line_to_rad(REAL sx, REAL sy, REAL ex, REAL ey, REAL *rad) ;
int  geo_ll(REAL *line1, REAL *line2, REAL *pnt) ;
REAL  geo_ln_ang_rad(REAL sx, REAL sy, REAL ex, REAL ey) ;
int geo_ln_p_arc(REAL xa, REAL ya, REAL xb, REAL yb, REAL xcen, REAL ycen, REAL
int  geo_ln_p_cir(REAL xa, REAL ya, REAL xb, REAL yb, REAL xcen, REAL ycen, REA
int  geo_ln_p_ln(REAL xa, REAL ya, REAL xb, REAL yb, REAL xc, REAL yc, REAL xd,
int  geo_ln_trim_pl(double  line[2][XY],                    int     flg,
int  geo_loop_r_or_l(int n, double pl[][XY], double *area) ;
void geo_mid_pnt(REAL xa, REAL ya, REAL xb, REAL yb, REAL *xmid, REAL *ymid) ;
void geo_mk_box(REAL p[XYZ], REAL size_x, REAL size_y, REAL pnt[][XYZ] ) ;
int geo_nv_ld(REAL sx, REAL sy,REAL ex, REAL ey,REAL dirx, REAL diry,REAL *vx,
int  geo_ofst_crgn(int nin, REAL pin[][XY], REAL offset, int *nout, REAL pout[]
int  geo_ofst_ll(REAL *l1, REAL *l2, REAL ofst, REAL pnt[XY]) ;
int geo_ofst_lnp(REAL sx, REAL sy, REAL ex, REAL ey,                    REAL dist
int  geo_ofst_lns(REAL sx, REAL sy, REAL ex, REAL ey, REAL dist, int side,
int  geo_ofst_orgn(int nin, REAL pin[][XY], REAL offset, int *nout, REAL pout[]
REAL geo_out_prod(REAL vx1, REAL vy1, REAL vx2, REAL vy2) ;
int geo_p_l_dist(REAL x, REAL y, REAL sx, REAL sy, REAL ex, REAL ey, REAL *dist
int  geo_perp_pnt(REAL sx,REAL sy,REAL ex,REAL ey,REAL x1,REAL y1,REAL *xprp,RE
int geo_pl3_to_bbox(int pln, int n, REAL pl[][XYZ], REAL bbox[MINMAX][2]) ;
int  geo_pl_to_bbox(int n, double pl[][XY], double bbox[MINMAX][XY]) ;
int geo_pnt_inbox(REAL xa, REAL ya, REAL xb, REAL yb, REAL xp, REAL yp) ;
int  geo_pnt_merg(REAL ipnt[][XY] , int *cnt ) ;
int geo_pnt_onarc(REAL xcen, REAL ycen, REAL rad, REAL sa, REAL ea, REAL xp, RE
int  geo_pnt_p_ln(REAL p[XY], REAL line[2][XY], REAL pnt[XY]) ;
```

実際、これしかない場合はココから探すしかないんですが。。

⑥ クラスの基本

◆ C++におけるクラスの書式

・ クラスの定義は、ヘッダファイル(.h)と実装ファイル(.cpp)に分ける

・ ファイル名には、クラス名の慣例である『C』はつけない

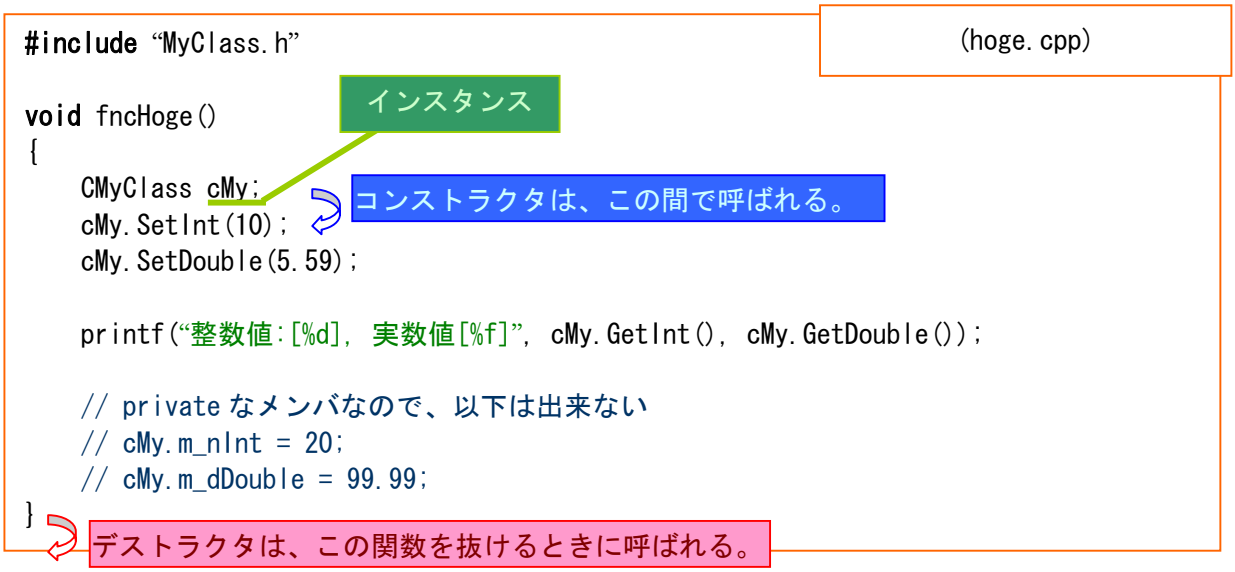
ヘッダファイル(MyClass.h)

```
class CMyClass{
public:
    CMyClass();                 // コンストラクタ
    ~CMyClass();                // デストラクタ

    void    SetInt(int val);
    void    SetDouble(double val);
    int     GetInt()const;
    double  GetDouble()const;   // メンバ関数

private:
    int m_nInt;
    double m_dDouble;           // メンバ変数
};
```

実装ファイル(MyClass.cpp)

```
CmyClass::CmyClass()
:   m_nInt(0)
,   m_dDouble(0.0)
{}

CMyClass::~CmyClass()
{}

void CmyClass::SetInt(int val)
{
    m_nInt = val;
}

void CmyClass::SetDouble(double val)
{
    m_dDouble = val;
}

int CmyClass::GetInt()const
{
    return m_nInt;
}

double CMyClass::GetDouble()const
{
    return m_dDouble;
}
```

・ 実際に使うときは、クラスのヘッダファイルをインクルードし、インスタンスを作って使う



- C 言語で、構造体につける名前に慣例として『St』をつけていたように、C++では慣例としてクラス名の先頭には『C』をつけることが多いです(別に決まりではないので、ついていなくてもかまいません)。
- 同様に、メンバ変数には『m_』をつける場合が多いです(好みにより分かれます)。
- 一般に、メンバ変数については『private』項目として定義し、外からは触れないようにします。
- クラスが持つ関数を、C++では『メンバ関数』と呼びます。ココからは、自分のメンバ変数を自由に参照できます。
- クラスの実体のことを『インスタンス』と呼びます。
- クラス名そのままのメンバ関数を『コンストラクタ』と呼びます。この関数は戻り値を持たない特別なメンバ関数で、インスタンスが生成されたときに必ず呼ばれます。
- メンバ関数を使うには、構造体のメンバ変数を参照するときと同じく、『クラス名.メンバ関数名』というように書きます。