オブジェクト指向と ゲームプログラミング

○　C++編 － 第３回　コンストラクタとデストラクタ　○

コンストラクタ

クラスには、コンストラクタ(constructor：構築子)という特別なメンバ関数があります。コンストラクタは、クラスと同じ名前を持ち、値を返さない(返せない)メンバ関数で、オブジェクトの生成時に自動的に呼び出されます。通常の関数のように引数を受け取り、オーバーロードで複数定義することができます。

コンストラクタは、名前どおり、オブジェクトを生成するための処理を記述する関数です。おもに、メンバ変数の初期化といったオブジェクトを生成するための適切な初期化作業を行います。コンストラクタの処理がすべて終わったとき、オブジェクトが完全に生成されます。コンストラクタをきちんと書くことで、オブジェクトの状態を生成時から保証することができるのです。

●コンストラクタの特徴
- クラス名と同じ名前を持つ
- オブジェクトがインスタンス化(実体化)したときに呼び出される
- 引数を受け取ることができる
- 戻り値を返すことができない
- オーバーロードできる

コンストラクタでオブジェクトを初期化する以外に、初期化のためのメンバ関数(仮にInit関数とします)を用意する方法も考えられます。この方法では、Init関数が呼び出されない限りオブジェクトの初期化が行われません。するとそのオブジェクトは、初期化されない限りまったく意味のないものとなり、他のメンバ関数が呼び出されたときの動作が予測できなくなってしまいます。このような理由から、オブジェクトの初期化はコンストラクタで行うようにします。

しかし、コンストラクタは戻り値を返せないなどの理由から、エラーが起きる可能性のある初期化作業(メモリの確保やDirectXなどのライブラリの初期化)は、初期化のための関数を作成した方が無難です。このような場合でも、コンストラクタで最低限の初期化作業は必要です。

前回のCharacterクラスにコンストラクタを追加してみましょう。

```
class Character {
public:
    // コンストラクタ
    Character(const double pos_x, const double pos_y);
    {
        m_PosX    = pos_x;
        m_PosY    = pos_y;
        m_AnimeNo = 0;
    }

    void Move(const double add_x, const double add_y);      // 移動
    void Animetion();                                       // アニメーション

private:
    double   m_PosX;        // x座標
    double   m_PosY;        // y座標
    int      m_AnimeNo;     // アニメーション番号
};
```

クラス内で定義されているCharacter関数がコンストラクタです。クラス名と同じ名前を持っており、voidなどの型の指定がありません。コンストラクタの定義はクラスの外に書いてもかまいません。クラスの外部に書く場合は、「Characterクラスのコンストラクタ」という意味で、「Character::Character」となることに注意してください。

このようにコンストラクタを書くと、以下のように、コンストラクタを呼び出してインスタンスを生成することができます。

```
Character  ply(310.3, 3.0);
```

この例では、plyオブジェクトの生成時にコンストラクタが呼び出され、メンバ変数m_PosX, m_PosYは引数で与えられた310.3と3.0、m_AnimeNoは0で初期化されます。

デフォルトコンストラクタ

前述のように引数を取るコンストラクタを１つでも定義した場合、

```
Character   ply;
```

のように、引数を与えずにオブジェクトを生成することができなくなってしまいます。これは、「データを与えずにオブジェクトを生成させない」という場合には便利な機能ですが、「データを与えなくてもオブジェクトを生成したい」という場合は、不便に感じます。このような場合は、以下のように引数を取らないコンストラクタを定義します。

```
class Character {
public:
    Character() { m_PosX = 0.0; m_PosY = 0.0; m_AnimeNo = 0; }      // コンストラクタ
    Character(const double pos_x, const double pos_y);              // コンストラクタ

    void Move(const double add_x, const double add_y);              // 移動
    void Animetion();                                               // アニメーション

private:
    double   m_PosX;        // ｘ座標
    double   m_PosY;        // ｙ座標
    int      m_AnimeNo;     // アニメーション番号
};
```

引数を取らないコンストラクタをデフォルトコンストラクタと呼びます。デフォルトコンストラクタにより、引数を与えないオブジェクトの生成が可能になります。

また、コンストラクタを１つも定義しない場合は、コンパイラによって自動的にコンストラクタが作成されるようになっています。これもデフォルトコンストラクタと呼びますが、その内容は何もしないというものです。メンバ変数の初期化などは行われないので、ほとんど役に立ちません。

コンストラクタ初期化子

コンストラクタでメンバ変数を初期化する際に、コンストラクタの中でメンバ変数に値を代入するのではなく、コンストラクタ初期化子で初期化することもできます。C++では、コンストラクタ初期化子でメンバ変数を初期化する方法が推奨されています。また、const指定されているメンバ変数を初期化できるのもこの方法だけです。

たとえば、Characterクラスの場合、コンストラクタでは３つのメンバ変数(m_PosX, m_PosY, m_AnimeNo)に初期値の代入を行います。これを、コンストラクタ初期化子を使ったものに変更すると、以下のようになります。

```
// デフォルトコンストラクタ
Character() : m_PosX(0.0), m_PosY(0.0), m_AnimeNo(0)
{
}

// コンストラクタ
Character::Character(const double pos_x, const double pos_y)
    : m_PosX(pos_x), m_PosY(pos_y), m_AnimeNo(0)
{
}
```

「:」の後ろにあるものがコンストラクタ初期化子です。このように書くと、メンバ変数が生成されると同時にカッコの中の値で初期化されます。Characterクラスの場合、コンストラクタ初期化子でメンバ変数を初期化すると、コンストラクタ本体で行うことがなくなるので、処理内容は空になります。

デストラクタ

クラスには、コンストラクタのほかに、デストラクタ(destructor：消去子)という特別なメンバ関数があります。これは、「~クラス名」という名前の特別な関数で、オブジェクトが破棄される(メモリから消えてなくなる)ときに、自動的に呼び出される関数です。デストラクタもコンストラクタと同様に戻り値がないので、先頭にvoidなど型の指定はできません。さらに、引数を受け取ることもできないので、オーバーロードして複数定義することもできません。

●デストラクタの特徴
・「~クラス名」という名前を持つ
・オブジェクトが破棄されるときに呼び出される
・引数を受け取ることができない
・オーバーロードできない
・戻り値を返すことはできない

デストラクタは、確保したメモリやライブラリの解放など、オブジェクトが破棄されるまでに必ず行わなければならない後始末を確実に行わせるための関数です。int型やdouble型のような、通常のメンバ変数は、オブジェクトが破棄されるときに自動的に破棄されるので、デストラクタでの解放処理は必要ありません。
後始末が不要な場合は、デストラクタを定義する必要はありませんが、後々クラスが継承される可能性もあるので、処理内容がなくても定義しておいた方が無難です。

Characterクラスのデストラクタは以下のようになります。特別な資源を使っていないので、デストラクタで行う処理はありません。

```
class Character {
public:
    // コンストラクタ
    Character() : m_PosX(0.0), m_PosY(0.0), m_AnimeNo(0) {}
    Character::Character(const double pos_x, const double pos_y)
        : m_PosX(pos_x), m_PosY(pos_y), m_AnimeNo(0) {}

    // デストラクタ
    ~Character() {}

    void Move(const double add_x, const double add_y);  // 移動
    void Animetion();                                   // アニメーション

private:
    double   m_PosX;        // x座標
    double   m_PosY;        // y座標
    int      m_AnimeNo;     // アニメーション番号
};
```

練習問題

1　以下のプログラムを入力し、コンストラクタとデストラクタがどのように実行されるかを確認しましょう。

```
#include <iostream>

using namespace std;

class CTest {
public:
    CTest()  { cout << "CTestのコンストラクタが呼ばれました" << endl; }
    ~CTest() { cout << "CTestのデストラクタが呼ばれました"   << endl; }
};

void main()
{
    cout << "*** main関数実行" << endl;
    CTest   test;
    cout << "*** testが生成されたました" << endl;
}
```

2　以下の文章の内容が正しい場合には○、間違っている場合には×を付けましょう。

(1)コンストラクタは、メンバ関数の一種である。
(2)コンストラクタの名前は、クラス名と同じでなければならない。
(3)コンストラクタは、戻り値を返せないので、戻り値のデータ型をvoidにしなければならない。
(4)引数を受け取るコンストラクタを定義することはできない。
(5)１つのクラスにつき１つのコンストラクタしか定義することができない。
(6)コンストラクタでは、メンバ変数の初期化を行うべきである。
(7)１つのクラスにつき１つのデストラクタしか定義することができない。

3　次のように整数値の座標を表すCPointクラスを作成しましょう。ただし、座標の範囲はｘが０～６４０、ｙが０～４８０となるようにしましょう。

メンバ変数
ｘ座標…pos_x
ｙ座標…pos_y

メンバ関数

```
void SetX(const int px);              // ｘ座標を設定する
void SetY(const int py);              // ｙ座標を設定する
int GetX();                           // ｘ座標を得る
int GetY();                           // ｙ座標を得る
```

コンストラクタ

```
CPoint();                             // ｘ，ｙ座標とも０で初期化する
CPoint(const int px, const int py);   // 初期座標を指定する(異常値の場合は０)
```

4　コンストラクタとデストラクタの利点を挙げましょう。

・コンストラクタの利点

・デストラクタの利点