Programming C++
Lecture Note 2
新しいクラスの定義の仕方

Jie Huang

クラスとクラスオブジェクト

- クラスの定義
  ここでは、鉄道車両の例でクラスの定義の仕方について説明する

```
class box_car{
  public:
    double height, width, length;
    double volume(){ return height * width * length;}
  protected:
    ...
  private:
    ...
};
```

- box_car:クラス名
  height, width, length:クラスメンバー変数
  volume:クラスメンバー関数
  public, protected, private:クラスの中の保護レベルを示す

クラスとクラスオブジェクト

- クラスのメンバーはpublic、privateとprotectedに分れる。
  - publicメンバーはクラスの内部と外部を問わずアクセスを許す。ここで、クラス内部とは主にクラスの内部関数からのアクセスを指す。
  - privateメンバーはクラス内部の関数からしかアクセスできない。
  - protectedメンバーはクラス内部関数からしかアクセスできないが、後に述べる派生クラスの関数からのアクセスを可能にするものである。

クラスとクラスオブジェクト

- クラスオブジェクトの生成
  - 実体を定義する
    ```
    box_car x;
    ```
  - ポインターを定義し、実体をnew演算子で確保する
    ```
    box_car *p = new box_car;
    ```
    ここで、newというキーワードは新しいオブジェクトのための領域を確保し、そのポインタを返す。
- クラスオブジェクトのメンバーの参照
  - クラスメンバーの参照は構造体と同じ方法を使う。
  - クラス演算子(.)による方法
    ```
    x.height = 2.5;
    ```
  - クラスポインタ演算子(->)を使う方法
    ```
    cout << p->volume();
    ```

クラスとクラスオブジェクト

- クラスメンバー関数の呼び出し
  - 普通のクラスメンバー関数はクラスオブジェクトのインスタンス(それぞれのクラスオブジェクト)に関連づけられるので、クラスオブジェクトのメンバーとして呼び出すことができる。

    ```
    box_car x, y;
    x.volume(); //オブジェクトxの容積を計算する
    y.volume(); //オブジェクトyの容積を計算する
    ```
  - クラスメンバー関数の中で使うメンバー変数は特に指定がなければ、呼び出す方のクラスオブジェクト(上の例ではxとy)に関連づけられる。
    - x.volume()から呼び出した時は関数内で使う変数height、width、lengthはそれぞれx.height、x.width、x.lengthと同じである。
    - y.volume()から呼び出した時は関数内で使う変数height、width、lengthはそれぞれy.height、y.width、y.lengthと同じである。

クラスとクラスオブジェクト

- クラスメンバー関数における暗黙の引数
  - クラスオブジェクト（例のx, y）はメンバー関数を呼び出すことによって直接関連づけられるので、引数として渡さなくてもメンバー関数からアクセスできる。つまり、メンバー関数への暗黙引数(implicit argument)としても解釈できる。
  - この暗黙の引数を明示的に示す場合は、thisポインタを使う。
  - 従って、これらのメンバー変数はメンバー関数内部においてはthis->height、this->width、this->lengthとしても記述できる。

クラスとクラスオブジェクト

- メンバー関数のプロトタイプと本体を分けて定義する方法
  - クラスが大きくなると、メンバー関数の定義を全部クラスの内部で行うのはプログラムを読みにくくするので、関数のプロトタイプだけをクラスの中に残し、本体をクラスの外部に置く方法がある。
  - メンバー関数のプロトタイプだけを含むクラス定義

```
class box_car{
  public:
    double height, width, length;
    double volume();     // 関数プロトタイプ宣言
};
```

  - スコープ演算子を使ったクラスメンバー関数の本体の定義

```
double box_car::volume(){ // 関数本体の定義
      return height * width * length;
}
```

ここで、”::”はスコープ演算子という。Volume関数がクラスbox_carのメンバー関数であることを示す。

クラスとクラスオブジェクト

- コンストラクタ(constructor)メンバー関数
  - クラスのコンストラクタはクラス名と同じ名前の関数で、クラスオブジェクトを生成する時に初期化の役割を担う。
  - コンストラクタは戻り値を指定する必要がない。
  - デフォルトのクラスコンストラクタ
    ```
    tank_car() { radius = 3.5; length = 40.0; }
    ```
  - パラメーター付きのコンストラクタ
    ```
    tank_car(double r, double l) { radius = r; length = l;}
    ```
  - デフォルト値付きパラメーターコンストラクタ
    ```
    tank_car(double r = 3.5, double l = 40.0) {
      radius = r; length = l;
    }
    ```
    呼び出す時に引数を省略した場合はデフォルト値を使うので、デフォルトコンストラクタとパラメーターコンストラクタの両方を兼ねることができる。

クラスとクラスオブジェクト

- デフォルトとパラメーターコンストラクタの定義

```
class tank_car{
  public:
    double radius, length;
    double volume(); // 関数プロトタイプの宣言
    tank_car(){ radius = 3.5; length = 40.0; }
    tank_car(double r, double l){ radius = r; length = l;}
};
```

- 両方を合わせた定義

```
class tank_car{
  public:
    double radius, length;
    double volume(); // 関数プロトタイプの宣言
    tank_car(double r = 3.5, double l = 40.0){
      radius = r; length = l;
    }
};
```

クラスとクラスオブジェクト

- クラスオブジェクトの生成とコンストラクタの呼び出し
  - コンストラクタはクラスオブジェクトがつくられる度に自動的に呼び出される。
  - 自動変数は関数がコールされる度に生成されるので、コンストラクタもその都度呼び出される。
  - 外部変数や静的変数はプログラムの開始時に一度だけ生成されるので、コンストラクタも一度だけ呼び出される。
  - new演算子で生成される場合は生成する時に呼び出される。
  - tank_car t1; // デフォルトの引数が使われる r=3.5, l=40.0
    tank_car t1(3.5, 40.0);
    tank_car t2(5, 50); //r=5.0, l=50.0 のオブジェクトを生成
    tank_car *tp = new tank_car(5, 50); //new演算子による生成

クラスとクラスオブジェクト

- クラスのインターフェース

```
class tank_car{
  private: double radius, length;
  public:
    tank_car(double r = 3.5, double l = 40.0){
      radius = r; length = l;
    }
    double read_radius(){ return radius;}
    double read_diameter(){ return radius * 2.0;}
    void write_radius(double r){ radius = r;}
    double volume(){ return pi * radius * radius * length; }
};
```

- ここで、read_radius、read_diameter、volumeなどの関数は内部データへのアクセスを提供するので、参照関数(reader)という。
- また、write_radiusなどの関数は内部データに値をセットするので、代入関数(writer)という。
- 参照と代入関数を合わせてアクセス関数(access function)という。

クラスとクラスオブジェクト

- データ保護とデータ抽象

```
class tank_car{
  private: double radius, length;
  public:
    tank_car(double r = 3.5, double l = 40.0){
      radius = r; length = l;
    }
    double read_radius(){ return radius;}
    double read_diameter(){ return radius * 2.0;}
    void write_radius(double r){ radius = r;}
    double volume(){ return pi * radius * radius * length;}
};
```

- 以上のクラスの定義で、クラスメンバー変数radius, lengthなどの内部データはクラスのプライベート部分にあるので、クラスの外部からは直接アクセスができない。
- この様に内部データを外部から隠蔽してしまい、そのアクセスをインタフェースを通してのみ提供することをデータ抽象(data abstraction)という。

クラスとクラスオブジェクト

- データ抽象(data abstraction)
  - アクセス関数は内部データの構造と必ずしも一致する必要がない。
  - データの抽象により、例えば内部データを以下のようにradiusからdiameter変更しても、インターフェースさえ変更しなければ外部からは全く変化は見られない。
  - ```
    class tank_car{
      private: double diameter, length;
      public:
        tank_car(double r = 3.5, double l = 40.0){
          diameter = r*2; length = l;
        }
        double read_radius(){ return diameter/2.0;}
        double read_diameter(){ return diameter;}
        void write_radius(double r){diameter = r*2.0;}
        double volume(){return pi*diameter*diameter*length/4.0;}
    };
    ```
    ```

クラスとクラスオブジェクト

- C++における構造体
  - C++の構造体は、特に指定がない場合、すべてのメンバーはpublicとして解釈される点以外はクラスとまったく同じである

```
struct Person {
    char  name[8];
    float height:
    float weight;
    int   sex;
};
Person taro;
```

  - Cと同じ記述もできる

```
struct Person taro;
```

  - これに対してクラスは特に指定しなければすべてのメンバーがprivateとなる。

プログラミングにおけるいくつかの注意

- ループが正しいかどうかをチェックするためのコツ
  - 目標の出力行数をrowsとする
  - 書く出した行数をrとする。
  - ループの中で常に書き出した行数をチェックする
  - // while文の前でチェック

```
int r = 0; //まだ1行も書いていないので0でなければならない
while (r != rows) {
   //ここはループの始めで、出力した行数がrであるとする
   std::cout <<...<< std::endl; //1行出力、rと出力行数が違う
   ++r; //rの値をインクリメント、出力行数とrの値が再び一致する
}
//ループの条件 r!=rows がfalseとなるので、
// r==rowsとなり、出力した行数がrowsであることを確認できる。
```

プログラミングにおけるいくつかの注意

- 半分開いた範囲
  例えば、[1, 3) は1と2を含むが、3は含まない。
- 入力データの最後をチェックする
  cin >> x; という文は変数xに値を代入するが、入力データがない場合はfalseを返すので、ループで以下の様に入力データの最後をチェックすることができる。

```
while(cin >>x) {
 /*ここで入力データの処理を行う*/
}
```

プログラム例

```
#include <iostream>
#include <string>
using std::cin;          using std::endl;
using std::cout;         using std::string;
int main(){
  cout << "Please enter your first name: ";
  string name; cin >> name;
  const string greeting = "Hello, " + name + "!";
  const int pad = 1; const int rows = pad * 2 + 3;
  const string::size_type cols = greeting.size() + pad * 2 + 2;
  cout << endl;
  ... // for-loop here
  return 0;
}
```

プログラム例

```
for (int r = 0; r != rows; ++r) {
  string::size_type c = 0;
  while (c != cols) {
    if (r == pad + 1 && c == pad + 1) {
      cout << greeting; c += greeting.size();
    } else {
      if (r == 0 || r == rows - 1 || c == 0
          || c == cols - 1) cout << "*";
      else cout << " ";
      ++c;
    }
  }
  cout << endl;
}
```

```
**********************
*//////////////////*
*//////////////////*
*//Hello, Taro!//*
*//////////////////*
*//////////////////*
**********************
```

空白部分を”/”で詰めて出力する