Programming C++
Lecture Note 1
オブジェクト指向言語とは

Jie Huang

コースの情報

- 担当教員: 黄　捷　(Jie Huang)
  - オフィス　126-B
  - メール　j-huang
  - 内線　3510
- 情報はwwwページにて提供
  - http://web-int/~j-huang/Lecture/ProgrammingII/prog2.html
- 関連授業
  - これまで: Programming 入門、Programming C、Programming Java I

授業のスタイル

- 成績評価：
  - クイズ：8～12回（講義始め15分程度、出席のチェックにも利用する）
  - 演習：13回分
  - 期末試験
  - 合格のためには**演習と試験の両方**を合格する必要がある。

演習問題の提出について

- 提出方法：演習担当教員の指示に従う
- 提出期限：次の授業の日（厳守）、期限過ぎた回答は評価しないか減点となります。
- 注意：プログラムを送るときは、名前と学籍番号を先頭においてください（コンパイルできるようにコメントアウトして）。プログラム以外の文章は入れないでください。 結果がある場合は最後にコメントアウトして付けてください。

コースの説明

- 教科書（参考書）
  - Accelerated C++, 効率的なプログラミングのための新しい定跡 (Japanese Edition)、訳、小林健一郎（ピアソンエデュケーション、2001）
  - トピック
    教科書に沿って教えるが、最初の何週間かでは教科書よりも早めにC++のクラスの知識について教える。

オブジェクト指向言語とは

- Objectとは
  簡単にいえば、Objectとはデータと関数（つまり、アルゴリズム、手続きとも言う）をひとまとめにした集合を言う。
- データと手続きをひとまとめにする利点
  - 手続きの対象となるデータがはっきりする。
  - データとアルゴリズムがまとまっているので、開発上管理しやすい。
  - 実世界のObject概念と共通する設計ができるので、高度複雑なシステムの設計がしやすい。

  例えば、ワールド時計というObjectは時間というデータを扱うことができ、機能として世界の各地域の現地時間を教えてくれる。

オブジェクト指向言語C++のスタイル

- Objectはclassというデータ構造によって実現される

```
class triangle { //三角形classの定義
  double x1, y1, x2, y2, x3, y3; //座標
public:
  double area() { //面積計算
    return (x1*y2+x2*y3+x3*y1
           -x1*y3-x2*y1-x3*y2)*0.5;
  }
};
triangle a; //三角形classのobjectの定義
a.area();     //変数aという三角形の面積計算
              // “.”はメンバー演算子という
```

オブジェクト指向言語C++のスタイル

- classは派生できる

```
class A { //基本クラスの定義
public:
  int va; //基本クラスの変数
  int fa(); //基本クラスの関数
};
class B : public A { //派生クラスの定義
public:
  int vb; //派生クラス固有の変数
  int fb(); //派生クラス固有の関数
};
```

この例では、クラスBはクラスAの派生クラスである。

- 派生クラス自身固有の変数と関数を持つほか、基底クラスの変数と関数を継承する。この例では、クラスBは変数va, vbと関数fa, fbを持つ。
- クラス派生と継承による差分プログラミング
すでに存在するプログラムを書き直さないで、サブクラスを定義して変更部分のみを新たに加える方法のことを言う。

オブジェクト指向言語C++のスタイル

- クラスのカプセル化とインターフェースの提供

```
structure point {double x; double y}; //座標点
class triangle { //座標構造体pointを利用した三角形クラスの定義
private: point p1, p2, p3; //内部データ部分
public: //インターフェース部分
  set_p1(point p){ p1 = p} //クラス変数に値を代入する関数
  set_p2(point p){ p2 = p} //write関数という
  set_p3(point p){ p3 = p}
  point get_p1(){ return p1;} //クラス変数の値を取得する関数
  point get_p2(){ return p2;} //read関数という
  point get_p3(){ return p3;} //合せてアクセス関数という
  double area(){ return(p1.x*p2.y+p2.x*p3.y+p3.x*p1.y
          -p1.x*p3.y-p2.x*p1.y-p3.x*p2.y)*0.5; }
};
```

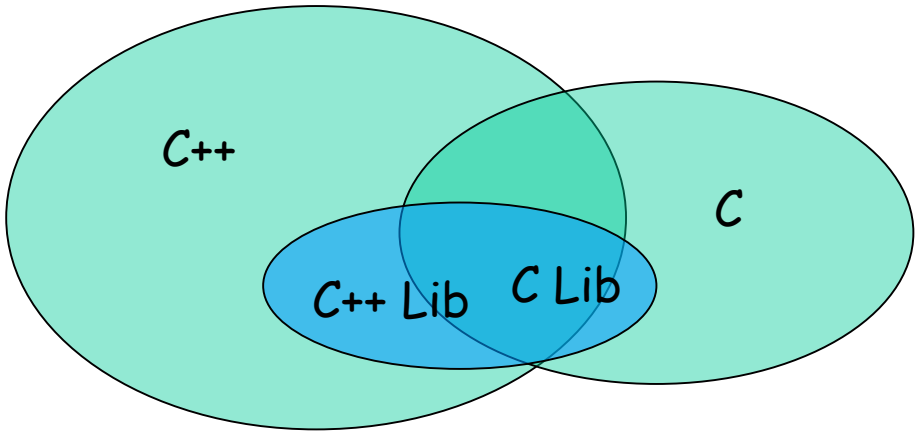
この場合、private部分にあるデータは外部からはアクセスできないので、**情報の保護**あるいは**カプセル化**と言う。また、その部分に対して用意するアクセス関数をインターフェースと言う。この時、クラスの持つ特性は内部の構造ではなく、インターフェースのみで定義できるので、**データの抽象化**という。

オブジェクト指向言語C++のスタイル

- オブジェクト指向言語の特徴
  - 手続き、関数中心のプログラムは処理のフローチャートに従ってプログラミングするので、データ構造は関数に付随するものである。
  - オブジェクト指向プログラムはデータと手続きあるいは関数が一体となっているので、データを中心にプログラミングを進めることができる。
- オブジェクト指向言語の利点
  - 保護されるデータがプログラムミスやバグなどによって不正に書き換えられたりしないので、安全性がある。
  - プログラムの仕様変更などが一部のソースコードだけの変更だけで対応できるので、柔軟性がある。
  - 同じソースコードを再利用したり、拡張したりしやすいので、再利用性がある。

C++標準ライブラリ

- C++とCの違い
  C++は元々Cのソースコードができるだけそのままコンパイル可能なように拡張されてきた。



C++標準ライブラリ

- ストリームクラスによる入出力例

```
#include <iostream>
using namespace std;
int main(){
  cout << "this is my first" << "C++ program";
  return 0;}
```

- namespaceライブラリやモジュールなどで複数の識別子（変数、関数など）をグループ化し、他のライブラリやプログラムのシンボルと衝突する危険を回避するためのもの。
- stdは標準ライブラリのnamespaceである。

```
using std::cout;
std::cout << "this is my first" << "C++ program";
```

などの使い方もある。C++標準ライブラリはすべての識別子をstdというネームスペースで定義している。

- 本コースは標準ライブラリのコンテナクラスの扱いを通して、C++言語について、その特徴、プログラミング技法などを学んでいく。

C++標準ライブラリ

- 出力演算子(output operator)<<
  挿入演算子(insertion operator)ともいう。
  cout << 表示したい値を与える式;
  cout << endl;　// 改行
  cout << １番目の式 << ２番目の式 << . . . << 最後の式;
- 入力演算子(input operator)>>
  抽出演算子(extraction operator)ともいう。
  cin >> 読み込んだ値が代入される変数
  cin >> １番目の変数 >> ２番目の変数 >> . . . >> 最後の変数
- iostream:標準入出力用ヘッダーファイル<>をつけて取り込む場合は“.h”は要らない。
  cout:標準出力
  cin:標準入力
  endl:改行
  string:文字列クラス
  size(): stringクラスのメンバー関数で、stringの長さを返す

C++標準ライブラリ

- 入出力演算子の定義方法
  分数を出力する出力演算子の例

```
ostream& operator<<(ostream& os, Fraction f)
{
  if (f.numerator == 0)
    os << “0”;              // 分子が0ならば0
  else if (f.denominator == 1)
    os << f.numerator; // 分母が1ならば、分子の値
  else
    os << f.numerator << “/” << f.denominator;
  return os;
}
```

- ostream:出力ストリームクラス
  operator:演算子を示すキーワード
  Fraction:別に定義した分数のクラス
  denominator:Fractionクラスの分母
  numerator:Fractionクラスの分子

簡単なプログラムの例

```
// ask for a person's name, and greet the person
#include <iostream>
#include <string>
int main(){
  std::cout << "please enter your first name: ";
  std::string name; std::cin >> name;
  std::cout << "Hello, " << name << "!"
            << std::endl;
  return 0;
}
```

ここで、cinとcoutはそれぞれ標準入力と出力で、endlは改行を出力する。
また、>>と<<はそれぞれ入力と出力演算子で、2つ以上続けて書くことで、連続した入出力ができる。
stringは文字列クラスで、C++標準ライブラリのクラスの1つ、使用に当たってはヘッダファイルstringをインクルードする必要がある。

フォーマットを工夫したプログラム

```
// ask for a person's name, and greet the person
#include <iostream>
#include <string>
int main(){
  std::cout << "please enter your first name: ";
  std::string name;  std::cin >> name; //標準入力から名前を入力
  const std::string greeting = "Hello, " + name + "!"; //あいさつ文
  const std::string spaces(greeting.size(),' '); //同じ長さのスペース
  const std::string second = "* " + spaces + " *";//前後に"*"をつける
  const std::string first(second.size(), '*'); //1行目の"*"の文字列
  std::cout << std::endl;
  std::cout << first << std::endl;
  std::cout << second << std:: endl;
  std::cout << "* " << greeting << " *" << std::endl;
  std::cout << second << std::endl;
  std::cout << first << std::endl;
  return 0;
}
```

フォーマットを工夫したプログラム

- 新しい文字列オブジェクトのいくつかの定義の方法について
  - 空の文字列を定義する
    ```
    std::string name;
    ```
  - 文字列を定義すると同時に初期化を行う
    ```
    std::string greeting = "Hello, " + name + "!";
    ```
  - 指定された長さの文字で初期化された文字列を定義する
    ```
    std::string spaces(greeting.size(), ' ');
    ```

プログラムの作成

- ソースファイルの拡張子は.ccとする
- コンパイルはg++, CCなど担当教員の指示に従う
- 標準入力（キーボード）から入力する代わりにファイルから入力する
  入力のリダイレクト
  プログラム名 < ファイル名
- スクリーンへの出力をファイルへリダイレクト
  プログラム名 > ファイル名