情報工学演習I
第１３回

C++の演習5（仮想関数）

授業の予定（後半）

#	月日	内容	担当者
7	11月13日	C言語の演習4 (ポインタの演算，列挙型)	内海
8	11月20日	C言語の演習課題	内海
9	11月27日	C++の演習1（クラス）	岩村
10	12月 4日	C++の演習2（クラスの継承）	岩村 (代理：谷川)
11	12月11日	C++の演習3 (関数のオーバロード)	岩村
12	12月18日	C++の演習4（インライン展開）	岩村
13	1月 8日	C++の演習5（仮想関数）	岩村
14	1月15日	C++の演習課題	谷川
15	1月22日	総合演習	谷川

今日の内容

- 第10回演習課題の解説
- 派生クラスへのポインタ
- 仮想関数
- 純粋仮想関数

第10回演習課題の解説

第１０回演習課題（１）

- 1. 以下の仕様を満たすプログラムを作れ
  - 以下の仕様を満たすクラスを持つ
    - 第９回演習課題（１）で作成したクラスを継承する
    - 数学、理科、英語に加えて、国語の点数を保存することができる
    - 合計４教科の点数を変更（上書き）できる関数と照会できる関数がある
    - 合計４教科の点数の平均を計算して返す関数がある
  - 上記のクラスを用いて、２人分のデータ（名前、４教科の点数）を順次コマンドラインから入力できる
  - 全員分の情報を入力した後、一人ずつ名前と平均点を表示する

第１０回演習課題（１）の回答

- ans10-1.cc
  - ans9-1_another.ccのクラスScoreをベースクラスにした
  - 気をつけるところ
    - class Score
      - データメンバのアクセス制限をprivatからprotectedに
    - class Score2（Scoreを継承したクラス）
      - 国語のスコアの宣言
      - 国語のスコアの参照を返すメンバ関数
      - メンバ関数average()を上書き
    - main関数
      - オブジェクトのクラスをScoreからScore2に
      - 国語のスコアを入力して、オブジェクトに渡すルーチン

main関数の変更箇所は
この2箇所だけ！

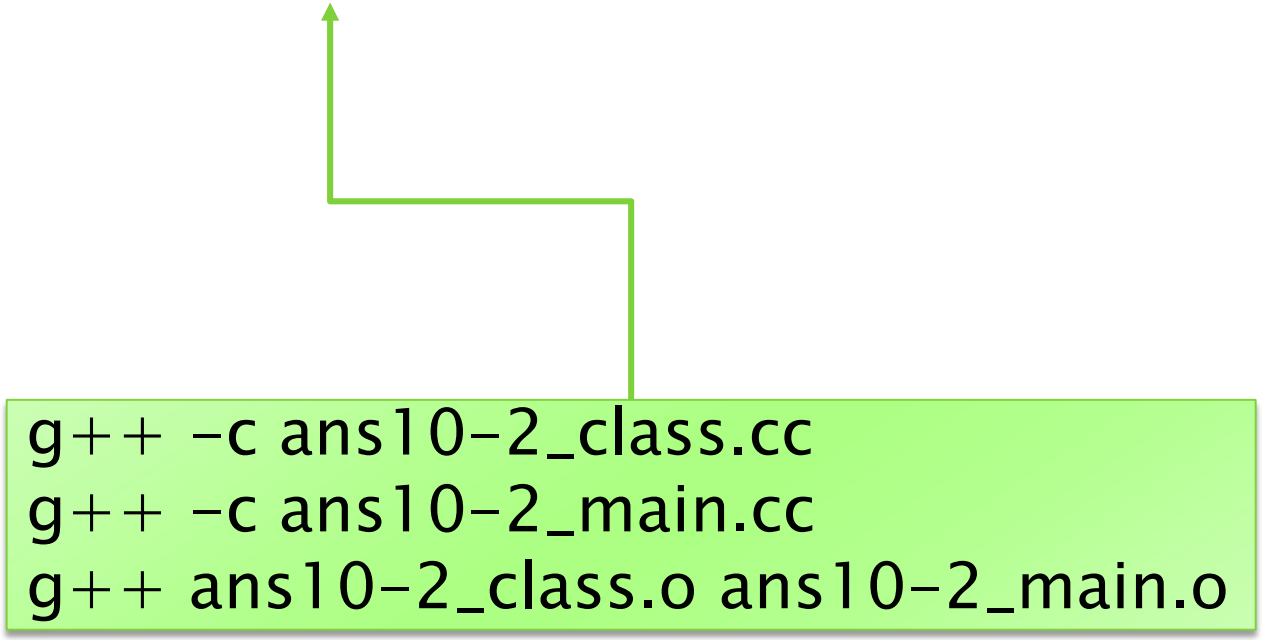
第１０回演習課題（２）

- 2. 以下の様に分割コンパイルを実現せよ
  - 第９回演習課題（２）で作成したプログラムを以下の3つに分割する
    - main関数を含むファイル
    - クラスのヘッダファイル
    - クラスのメンバ関数の定義を含むファイル
  - コンパイルするコマンドをテキストファイルに書く
    - 例：

      g++ -c ex5_main.cc
      g++ -c ex5_coordinate.cc
      g++ ex5_main.o ex5_coordinate.o
  - ソースファイル（３つ）とコンパイルするコマンドを提出

第１０回演習課題（２）の回答

- ans9-2_another.ccを3分割した
  - クラス定義：ans10-2_class.cc, ans10-2_class.h
  - main関数：ans10-2_main.cc
  - コンパイルの手順：ans10-2_command



第１０回演習課題（３）

- 3. 以下のプログラムを作成する
  - 引数を2つ取る
    - １つはファイル名
    - もう一つは、キーワード
  - ファイルの中にキーワードが何回出現するかを数える

第１０回演習課題（３）の回答

やり方はひとつではない

- ans10-3.cc
  - 方針
    - C言語の知識で実装
      - getlineコマンドでファイルからテキストを1行ずつ読み込む
      - strstrコマンドでテキスト（1行）ずつキーワードと比較
        - もしキーワードが見つからなければ、次の行へ
        - もしキーワードが見つかれば、見つかった箇所までのテキストを無視して再度比較する。これを見つからなくなるまで繰り返す。

getline：ファイルから1行読み込む
(http://linuxjm.sourceforge.jp/html/LDP_man-pages/man3/getline.3.html)

strstr：文字列１から文字列２を検索する
(http://www9.plala.or.jp/sgwr-t/lib/strstr.html)

派生クラスへのポインタ

別クラスへのポインタ

- 通常は、あるクラスのポインタが異なるクラスのポインタを指すことはできない

ex19_pointer_test.cc

別クラスへのポインタ

```
#include <iostream>
using namespace std;

class a {};
class b {};

int main () {
  a obja; // クラスaのオブジェクト
  b objb; // クラスbのオブジェクト

  a *pa = &obja; // クラスaのオブジェクトへのポインタ
  b *pb = &objb; // クラスbのオブジェクトへのポインタ

  // a *pa2 = &objb; // クラスaのポインタでクラスbを指す場合(エラーになる)
}
```

派生クラスへのポインタ

- 基本クラスのポインタを使って派生クラスにアクセスできる
- ただし、派生クラスにだけ存在するメンバ関数にはアクセスできない
- 仮想関数を使うときに意味を持つ

派生クラスへのポインタ

ex19_derive_pointer.cc

```
#include <iostream>
using namespace std;

class base { // 基本クラス
public:
  int i;
  base(int x) { // コンストラクタ(値の代入)
    i = x;
  }
  void tasu(int x) { // 足す
    cout << i + x << endl;
  }
};
```

```
class deriv : public base { // 派生クラス
public:
  deriv(int x) : base(x) {} // コンストラクタが呼ばれれば、基本クラスのコンストラクタで初期化

  void kakeru(int x) { // 掛ける
    cout << i * x << endl;
  }
};
```

ex19_derive_pointer.cc

派生クラスへのポインタ 続き

```
int main() {
  base *p; // クラスbaseへのポインタ
  base b(10); // クラスbaseのオブジェクト
  deriv d(10); // クラスderivのオブジェクト

  p = &b; // ポインタはクラスbaseのオブジェクトを指す
  p->tasu(5); // baseのtasu()を使用する

  p = &d; // ポインタはクラスderivのオブジェクトを指す
  p->tasu(5); // deriv1のtasu()を使用する(実際はbaseのtasu()を使用)
  // p->kakeru(5); // deriv1のkakeru()を使用する(エラーになる)

  return 0;
}
```

仮想関数

仮想関数

- 基本クラス内で宣言され、派生クラス内で再定義されるメンバ関数

仮想関数

ex20_virtual.cc

```
#include <iostream>
using namespace std;

class base { // 基本クラス
public:
  int i;
  base(int x) { // コンストラクタ
    i = x;
  }

  virtual void func() { // 仮想関数
    cout << "baseのfunc()を使う: ";
    cout << i << endl;
  }
};
```

```
// 派生クラス1
class deriv1 : public base {
public:
  deriv1(int x) : base(x) {} // コンストラクタ：基本クラスのコンストラクタで初期化

  void func() {
    cout << "deriv1のfunc()を使う: ";
    cout << i*i << endl;
  }
};
```

仮想関数 続き

ex20_virtual.cc

```
// 派生クラス2
class deriv2 : public base {
public:
  deriv2(int x) : base(x) {} // コンストラクタ：基本クラスのコンストラクタで初期化

  void func() {
    cout << "deriv2のfunc()を使う: ";
    cout << i+i << endl;
  }
};
```

仮想関数 続き

ex20_virtual.cc

```
int main() {
  base *p; // クラスbase型のポインタ
  base b(10); // クラスbaseのオブジェクト
  deriv1 d1(10); // クラスderiv1のオブジェクト
  deriv2 d2(10); // クラスderiv2のオブジェクト

  p = &b; // ポインタはクラスbaseのオブジェクトを指す
  p->func(); // baseのfunc()を使用する
  p = &d1; // ポインタはクラスderiv1のオブジェクトを指す
  p->func(); // deriv1のfunc()を使用する
  p = &d2; // ポインタはクラスderiv2のオブジェクトを指す
  p->func(); // deriv2のfunc()を使用する

  return 0;
}
```

同じ関数を3回呼び出している
（ように見える）

実行例

- p->func();を3回実行したが、結果は違う

```
$ ./a.exe
baseのfunc()を使う: 10
deriv1のfunc()を使う: 100
deriv2のfunc()を使う: 20
```

- 試しに12行目のvirtualを除いてみたらこうなる

```
$ ./a.exe
baseのfunc()を使う: 10
baseのfunc()を使う: 10
baseのfunc()を使う: 10
```

仮想関数

- 仮想関数は継承したクラスで上書きできる

クラスbase

```
virtual void func() { // 仮想関数
  cout << "baseのfunc()を使う: ";
  cout << i << endl;
}
```

上書き

クラスderiv1

```
void func() {
  cout << "deriv1のfunc()を使う: ";
  cout << i*i << endl;
}
```

クラスderiv2

```
oid func() {
 cout << "deriv2のfunc()を使う: ";
 cout << i+i << endl;
}
```

仮想関数

- 関数のオーバーロードとの違い
  - 関数のオーバーロードでは、引数の型が違う必要がある
  - 仮想関数では、引数の型は同じ

純粋仮想関数

純粋仮想関数

- 具体的な関数の中身を定義せずに、引数と戻り値のみが定義された仮想関数
- 関数の中身は派生クラスで定義する必要がある
- 書式

```
virtual type func-name( parameter-list) = 0;
```

- 抽象クラス
  - 少なくとも１つの純粋仮想関数を含むクラス
  - 不完全なクラスであり、オブジェクトを作成できない

純粋仮想関数

ex21_pure_virtual.cc

```
#include <iostream>
using namespace std;

class keisan { // 基本クラス
public:
  int i;

  keisan(int x) { // コンストラクタ
    i = x;
  }

  virtual void func(int x) = 0; // 純粋仮想関数
};
```

```
class tasu : public keisan { // 派生クラス1(足し算)
public:
  tasu(int x) : keisan(x) {} // コンストラクタが呼ばれれば、基本クラスのコンストラクタで初期化

  void func(int x) {
    cout << i + x << endl;
  }
};
```

純粋仮想関数

ex21_pure_virtual.cc

```
class kakeru : public keisan
{ // 派生クラス2(かけ算)
public:
  kakeru(int x) : keisan(x) {} // コンストラクタが呼ばれれば、基本クラスのコンストラクタで初期化


  void func(int x) {
    cout << i * x << endl;
  }
};
```

```
int main() {
  // keisan b(10); // クラスkeisanのオブジェクト(エラー)
  tasu t(10); // クラスtasuのオブジェクト
  kakeru k(10); // クラスkakeruのオブジェクト

 keisan *p; // クラスkeisan型のポインタ

  p = &t; // ポインタはクラスtasuのオブジェクトを指す
  p->func(5); // tasuのfunc()を使用する

  p = &k; // ポインタはクラスkakeruのオブジェクトを指す
  p->func(5); // kakeruのfunc()を使用する

  return 0;
}
```

演習課題

第１３回演習課題（１）

- 1. 任意の数nをキーボードから入力して、その数nの階乗n!の桁数を求めるプログラムを作成せよ。そして、横軸：n、縦軸：n!の桁数のグラフを書け（nは1から100とする）。ただし、桁数は近似で良い。

第１３回演習課題（２）

- 2. プログラムを作り、次の問いを解け
  - 左右どちらから読んでも同じ値になる数を回文数という
  - 2桁の数の積で表される回文数のうち、最大のものは 9009 = 91 × 99 である
  - では、3桁の数の積で表される回文数の最大値を求めよ

提出に関して

- 提出するもの
  - ソースファイル(.ccまたは.cpp ファイル)
    - ファイル名はkadai0108_学籍番号_課題番号.cc（.cpp）
    - (Visual Studioの場合)
      ファイル名はkadai0108_学籍番号_課題番号_v.cc
      （.cpp）
  - 実行結果の出力と講義に関するコメント
    - .txt ファイルで、学籍番号、氏名を含む
    - ファイル名はreport0108_学籍番号.txt とする

提出に関して（続き）

- 提出期限
  - 1月22日（水）　００：００
- 提出方法
  - 授業支援システムから提出
- 注意点
  - ファイル名の命名規則が間違っているものは採点しない
  - コンパイルの通らないものは採点しない