2023 年度 システム構成論レポート

学修番号: 22140003
氏名: 佐倉仙汰郎

レポート提出日：2024/1/19

課題 1

授業で示した例以外で，基底クラス，派生クラスを自由に作成しなさい．また，作成したクラスに対して仮想関数を定義し，その機能を確認しなさい．

Listing 1　kadai1

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

class character{
    private:
    string name_;
    public:
    character(string name) : name_(name) {}

    virtual void print_info() const = 0;
};

class Archer : public character{
    private:
    int hp = 100;
    int attack = 30;
    string job = "Archer";
    public:
    Archer(string name) : character(name) {}
    void print_info() const override{
        cout << "job :" << job << endl;
        cout << "hp :" << hp << endl;
        cout << "attack :" << attack << endl;
    }
};

class Warrior : public character{
    private:
    int hp = 250;
    int attack = 10;
    string job = "Warrior";
    public:
    Warrior(string name) : character(name) {}
    void print_info() const override{
        cout << "job :" << job << endl;
        cout << "hp :" << hp << endl;
        cout << "attack :" << attack << endl;
```

```
    }
};



int main(){
    Archer A("Satoshi");
    Warrior B("Yui");

    A.print_info();
    B.print_info();
    return 0;

}
```

出力例

```
job :Archer
hp :100
attack :30
job :Warrior
hp :250
attack :10
```

課題 2

授業で示した複素数クラス COMPLEX の加算以外の四則演算 (減算，乗算，除算) を実行する演算子関数を作成しなさい．複素数 a, b に対して，a += b (= a + b) を実行する演算子関数 operator+=を作成しなさい.

Listing 2　kadai2

```
#include<iostream>

using namespace std;

class COMPLEX
{
private:
double re_;
double im_;
public:
COMPLEX(double re = 0, double im = 0) : re_{re}, im_{im} {}
double re() { return re_; }
double im() { return im_; }
friend COMPLEX operator+(const COMPLEX&, const COMPLEX&);
friend COMPLEX operator−(const COMPLEX&, const COMPLEX&);
friend COMPLEX operator∗(const COMPLEX&, const COMPLEX&);
friend COMPLEX operator/(const COMPLEX&, const COMPLEX&);
COMPLEX& operator+=(const COMPLEX&);
COMPLEX conjugate();
friend ostream& operator<<(ostream &os, const COMPLEX& c);
};



COMPLEX operator−(const COMPLEX& X, const COMPLEX& Y){
    double r = X.re_ − Y.re_;
    double i = X.im_ − Y.im_;
    COMPLEX Z(r,i);
    return Z;
}

COMPLEX operator∗(const COMPLEX& X, const COMPLEX& Y){
    double r = (X.re_∗Y.re_) − (X.im_∗Y.im_);
    double i = (X.im_∗Y.re_) + (X.re_∗Y.im_);
    COMPLEX Z(r,i);
    return Z;
}
```

```
COMPLEX operator/(const COMPLEX& X, const COMPLEX& Y){
    double r = ((X.re_*Y.re_) + (X.im_*Y.im_)) / ((Y.im_*Y.im_) + (Y.re_*Y.re_));
    double i = ((−X.re_*Y.im_) + (X.im_*Y.re_)) / ((Y.im_*Y.im_) + (Y.re_*Y.re_));
    COMPLEX Z(r,i);
    return Z;
}


int main(){

    COMPLEX x(3,10);
    COMPLEX y(1,1);
    //minus
    COMPLEX z = x − y;
    cout << z.re() << " + " << z.im() << "i" << endl;

    //multiple
    z = x * y;
    cout << z.re() << " + " << z.im() << "i" << endl;

    //devide
    z = x / y;
    cout << z.re() << " + " << z.im() << "i" << endl;

    return 0;
}
```

出力例

```
2 + 9i
-7 + 13i
6.5 + 3.5i
```

課題 3

次式で定義される関数 f(x) の値を返す関数オブジェクトを作成しなさい．以下の積分を台形近似により求める関数を作成し，$a = 1, x \to \infty$ に対して計算例を示しなさい (フレネル積分).

Listing 3　kadai3

```
#include<iostream>
#include<cmath>

using namespace std;

class f{
    private:
    double a_;
    public:
    f(double p) : a_{p} {}
    ~f() {}

    double operator()(const double &x) const {
        return cos(a_ * x * x);
    }
};

double func(double x){
    double a = 1.0;
    return cos(a * x * x);
}

double fresnel(double (*F)(double)){
    double sum = 0;
    double gosa = 0.01;
    double d = 0.01;
    double x0 = 0;
    int count = 0;
    while(true){
        double X = (F(x0) + F(x0 + d));
        x0 += d;
        sum += X;
        count ++;
        if(count == 1000)break;
        cout << X << endl;
    }
    return sum;
}
```

```
int main(){
    cout << fresnel(func) << endl;
    return 0;
}
```

出力例

```
none
```

課題 4

整数，実数，string 型等，大小関係が定義されている任意の型 T のオブジェクト a，b を入力とする．a と b のうち大きい方を返す関数を関数テンプレートにより作成し，その動作を確認しなさい．また，この関数テンプレートが動作するクラスを 1 つ定義し，その動作を確認しなさい．

Listing 4　kadai4

```
#include <iostream>
#include <string>

template <typename T>
T max_value(const T& a, const T& b) {
    return (a > b) ? a : b;
}

template <typename T>
class ExampleClass {
public:
    ExampleClass(const T& val) : value(val) {}

    T getValue() const {
        return value;
    }

    ExampleClass<T> max(const ExampleClass<T>& other) const {
        return ExampleClass<T>(max_value(value, other.getValue()));
    }

private:
    T value;
};

int main() {
    int int_a = 5, int_b = 8;
    std::cout << max_value(int_a, int_b) << std::endl;

    double double_a = 3.14, double_b = 2.71;
    std::cout << max_value(double_a, double_b) << std::endl;

    std::string str_a = "apple", str_b = "banana";
    std::cout << max_value(str_a, str_b) << std::endl;

    ExampleClass<int> int_instance_a(10), int_instance_b(15);
    ExampleClass<int> max_int_instance = int_instance_a.max(int_instance_b);
```

```
    std::cout << max_int_instance.getValue() << std::endl;

    ExampleClass<std::string> str_instance_a("cat"), str_instance_b("dog");
    ExampleClass<std::string> max_str_instance = str_instance_a.max(str_instance_b);
    std::cout << max_str_instance.getValue() << std::endl;

    return 0;
}
```

出力例

```
Max of integers: 8
Max of doubles: 3.14
Max of strings: banana
Max of ExampleClass<int>: 15
Max of ExampleClass<std::string>: dog
```