Министерство образования и науки РФ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования

Тульский государственный университет

КАФЕДРА АВТОМАТИКИ И ТЕЛЕМЕХАНИКИ

НАСЛЕДОВАНИЕ. ВИРТУАЛЬНЫЕ ФУНКЦИИ.

Лабораторная работа № 4
по курсу «Объектно-ориентированное программирование»

Вариант № 11

Выполнил: студент группы 220601 _____ Белым А.А.
(подпись)
Проверил: к. ф.-м. н., доцент каф. АТМ _____ Середин О.С.
(подпись)

Тула 2013

Цель работы

Изучить принцип наследования. Понять назначение виртуальных функций. Написать программу на C++ с их использованием.

Задание

Задан класс:

```
class CMPLX
 {
   public:
      CMPLX();  /*1  Инициализация как (0,0) */
      CMPLX (float real, float imag);  /*2  Инициализация как (real,imag) */
      void Setcompl(float, float);  /*3  Изменение числа */
      void Inc();  /*4  Инкремент: |z|=|z|+1  */
      void Dec();   /*5  Декремент: |z|=|z|-1  */
      CMPLX Add(CMPLX what);  /*6  Сложение */
      CMPLX Sub(CMPLX what);  /*7  Вычитание */
      CMPLX Mul(CMPLX what);  /*8  Умножение */
      CMPLX Cmul(float what);  /*9  Умножение на действительное число
      */
      CMPLX Div(CMPLX what);  /*10  Деление */
      void Print();  /*11  Вывод в виде “(“re”,” im”)” */
    private:
          float re;
          float im;
  }
```

Реализовать следующие методы класса и написать демонстрационную программу, инициализирующую несколько переменных посредством конструктора №2, выводящую их, производящую над ними некоторые действия с помощью методов №№4,7,11; и выводящую результаты.

Используя данный класс, объявить в нем функции, соответствующие номеру варианта, виртуальными и дополнить программу классом, в котором будет использованы эти функции. Написать демонстрационную программу использования данных виртуальных функций.

Теоретическая справка

Реализация класса

Ниже представлено определение базового класса «число», файл NUMBER.h:

```
#ifndef NUMBER_H
#define NUMBER_H
class CMPLX;
//!Абстрактный класс "число".
class NUMBER{
  public:
    /*!Инкремент - увеличение числа на единицу.*/
    virtual void Inc()=0;
    /*!Декремент - уменьшение числа на единицу.*/
    virtual void Dec()=0;
    /*!Операция сложения с комплексным числом.
     * \param what слагаемое
     * \return сумму комплексных чисел
     */
    virtual CMPLX Add(CMPLX what) const=0;
    /*!Операция вычитания комплексного числа.
     * \param what вычитаемое
     * \return разность комплексных чисел
     */
    virtual CMPLX Mul(CMPLX what) const=0;
    /*!Операция умножения на комплексное число.
     * \param what множитель
     * \return результат умножения
     */
    virtual CMPLX Div(CMPLX what) const=0;
    /*!Операция деления на комплексное число.
     * \param what делитель
     * \return результат деления
     */
    virtual CMPLX Sub(CMPLX what) const=0;
    /*!Вывод числа на экран.*/
    virtual void Print() const=0;
    /*!Оператор приведения к комплексному числу.*/
    virtual operator CMPLX() const=0;
};
#include "CMPLX.h"
#endif //NUMBER_H
```

Ниже представлено определение класса «комплексное число», файл CMPLX.h:

```
#ifndef CMPLX_HPP
#define CMPLX_HPP
#include <iostream>
#include "NUMBER.h"
/*!Класс комплексных чисел.*/
class CMPLX:public NUMBER {
    public:
        /*!Конструктор по умолчанию.*/
        CMPLX();
        /*!Конструктор, задающий вещественную и мнимую части.
         * \param real вещественная часть
         * \param imag мнимая часть
         */
        CMPLX(float real,float imag);
        /*!Установка значения.
         * \param real вещественная часть
         * \param imag мнимая часть
         */
        void Setcompl(float real,float imag);
        /*!Инкремент - увеличение модуля на единицу.*/
```

```
        void Inc();
        /*!Декремент - уменьшение модуля на единицу.*/
        void Dec();
        /*!Операция сложения комплексных чисел.
         * \param what второе слагаемое
         * \return сумму комплексных чисел
         */
        CMPLX Add(CMPLX what) const;
        /*!Операция вычитания комплексных чисел.
         * \param what вычитаемое
         * \return разность комплексных чисел
         */
        CMPLX Sub(CMPLX what) const;
        /*!Операция умножения комплексных чисел.
         * \param what множитель
         * \return результат умножения
         */
        CMPLX Mul(CMPLX what) const;
        /*!Операция умножения на вещественное число.
         * \param what множитель
         * \return результат умножения
         */
        CMPLX Cmul(float what) const;
        /*!Операция деления комплексных чисел.
         * \param what делитель
         * \return результат деления
         */
        CMPLX Div(CMPLX what) const;
        /*!Вывод числа на экран в формате (real,imag).*/
        void Print() const;
        /*!Оператор приведения к комплексному числу.*/
        operator CMPLX() const;
    private:
        //!Вещественная часть
        float re;
        //!Мнимая часть
        float im;
};
#endif //CMPLX_HPP
```

Ниже представлена реализация класса «комплексное число», файл CMPLX.cpp:

```
#include <iostream>
#include <cmath>
#include "CMPLX.h"
#include "REAL.h"
using namespace std;
CMPLX::CMPLX():re(0),im(0){
}

CMPLX::CMPLX(float real,float imag):re(real),im(imag){
}

void CMPLX::Setcompl(float r,float i){
    re=r; im=i;
}

void CMPLX::Inc(){
    float r=sqrt(re*re+im*im)+1,phi=atan2(im,re);
    re=r*cos(phi); im=r*sin(phi);
}

void CMPLX::Dec(){
    float r=sqrt(re*re+im*im)-1,phi=atan2(im,re);
```

```
    re=r*cos(phi); im=r*sin(phi);
}

CMPLX CMPLX::Add(CMPLX what) const {
    CMPLX temp;
    temp.re=re+what.re;
    temp.im=im+what.im;
    return temp;
}

CMPLX CMPLX::Sub(CMPLX what) const {
    CMPLX temp;
    temp.re=re-what.re;
    temp.im=im-what.im;
    return temp;
}

CMPLX CMPLX::Mul(CMPLX what) const {
    CMPLX temp;
    temp.re=re*what.re-im*what.im;
    temp.im=im*what.re+re*what.im;
    return temp;
}

CMPLX CMPLX::Cmul(float what) const {
    CMPLX temp;
    temp.re=re*what;
    temp.im=im*what;
    return temp;
}

CMPLX CMPLX::Div(CMPLX what) const {
    CMPLX temp;
    float denom=what.re*what.re+what.im*what.im;
    temp.re=(re*what.re+im*what.im)/denom;
    temp.im=(im*what.re-re*what.im)/denom;
    return temp;
}

void CMPLX::Print() const {
    std::cout<<"("<<re<<","<<im<<")"<<std::endl;
}

CMPLX::operator CMPLX() const {
    return *this;
}
```

Ниже представлено определение класса «вещественное число», файл REAL.h:

```
#ifndef REAL_H
#define REAL_H
#include "NUMBER.h"
#include "CMPLX.h"
/*!Класс "вещественное число".*/
class REAL:public NUMBER{
  private:
    double val;//!<значение числа
  public:
    /*!Конструктор по умолчанию.*/
    REAL();
    /*!Конструктор, задающий значение числа.
     * \param value значение числа
     */
```

```
    REAL(double value);
    /*!Получение значения числа.
     * \return значение числа
     */
    double value() const;
    /*!Установка значения числа.
     * \param value новое значение числа
     */
    void setValue(const double value);
    /*!Инкремент - увеличение числа на единицу.*/
    void Inc();
    /*!Декремент - уменьшение числа на единицу.*/
    void Dec();
    /*!Операция сложения с комплексным числом.
     * \param what слагаемое
     * \return сумму комплексных чисел
     */
    CMPLX Add(CMPLX what) const;
    /*!Операция умножения на комплексное число.
     * \param what множитель
     * \return результат умножения
     */
    CMPLX Mul(CMPLX what) const;
    /*!Операция деления на комплексное число.
     * \param what делитель
     * \return результат деления
     */
    CMPLX Div(CMPLX what) const;
    /*!Операция вычитания комплексного числа.
     * \param what вычитаемое
     * \return разность комплексных чисел
     */
    CMPLX Sub(CMPLX what) const;
    /*!Вывод числа на экран.*/
    void Print() const;
    /*!Оператор приведения к комплексному числу.*/
    operator CMPLX() const;
};
#endif //REAL_H
```

Ниже представлена реализация класса «вещественное число», файл REAL.cpp:

```
#include "REAL.h"
#include <iostream>
using namespace std;
REAL::REAL():val(0){}

REAL::REAL(double value):val(value){};

double REAL::value() const{
    return val;
};

void REAL::setValue(const double value){
    val=value;
}
void REAL::Inc(){
    ++val;
}
void REAL::Dec(){
    --val;
}
void REAL::Print() const {
    std::cout<<val<<std::endl;
```

```
}
CMPLX REAL::Sub(CMPLX what) const {
    CMPLX c=CMPLX(val,0);
    return c.Sub(what);
}
CMPLX REAL::Add(CMPLX what) const {
    CMPLX c=CMPLX(val,0);
    return c.Add(what);
}
CMPLX REAL::Mul(CMPLX what) const {
    CMPLX c=CMPLX(val,0);
    return c.Mul(what);
}
CMPLX REAL::Div(CMPLX what) const {
    CMPLX c=CMPLX(val,0);
    return c.Div(what);
}
REAL::operator CMPLX() const {
    return CMPLX(this->val,0);
}
```

Демонстрационная программа

Далее приводится демонстрационная программа, файл lab4.cpp:

```
#include "NUMBER.h"
#include "CMPLX.h"
#include "REAL.h"
#include <iostream>
using namespace std;
void doFunWithNumbers(NUMBER &a,NUMBER &b){
    cout<<"a=";
    a.Print();
    cout<<"b=";
    b.Print();
    cout<<"a.Sub(b)=";
    const NUMBER& c=a.Sub(b);
    c.Print();
    cout<<"a.Inc()=";
    a.Inc();
    a.Print();
    cout<<"b.Inc()=";
    b.Inc();
    b.Print();
}

int main(){
    double r,i;
    cout<<"Введите вещественную и мнимую части комплексного числа."<<endl;
    cin>>r>>i;
    CMPLX a(r,i);
    cout<<"Введите вещественное число."<<endl;
    cin>>r;
    REAL b(r);
    doFunWithNumbers(a,b);
    doFunWithNumbers(b,a);
    return 0;
}
```

Далее приводится описание функций, методов, типов данных и классов.

Класс NUMBER

Абстрактный класс "число".

Является базовым для классов **CMPLX** и **REAL**.

```
#include <NUMBER.h>
```

Открытые члены

- virtual void **Inc** ()=0
- virtual void **Dec** ()=0
- virtual **CMPLX Add** (**CMPLX** what) const =0
- virtual **CMPLX Mul** (**CMPLX** what) const =0
- virtual **CMPLX Div** (**CMPLX** what) const =0
- virtual **CMPLX Sub** (**CMPLX** what) const =0
- virtual void **Print** () const =0
- virtual **operator CMPLX** () const =0

Методы

virtual CMPLX NUMBER::Add (CMPLX *what*) const [pure virtual]

Операция сложения с комплексным числом.

Аргументы метода представлены в таблице 1.

Аргументы:

Таблица 1 – Аргументы метода сложения

what	слагаемое

Возвращает:

сумму комплексных чисел

Замещается в **REAL::Add** и **CMPLX::Add**.

virtual void NUMBER::Dec () [pure virtual]

Декремент - уменьшение числа на единицу.

Замещается в **REAL::Dec** и **CMPLX::Dec**.

virtual CMPLX NUMBER::Div (CMPLX *what*) const [pure virtual]

Операция умножения на комплексное число.

Аргументы метода представлены в таблице 2.

Аргументы:

Таблица 2 – Аргументы метода деления

what	множитель

Возвращает:

результат умножения

Замещается в **CMPLX::Div** и **REAL::Div**.

virtual void NUMBER::Inc () `[pure virtual]`

Инкремент - увеличение числа на единицу.

Замещается в **REAL::Inc** и **CMPLX::Inc**.

virtual CMPLX NUMBER::Mul (CMPLX *what*) const `[pure virtual]`

Операция вычитания комплексного числа.

Аргументы метода представлены в таблице 3.

Аргументы:

Таблица 3 – Аргументы метода умножения

what	вычитаемое

Возвращает:

разность комплексных чисел

Замещается в **CMPLX::Mul** и **REAL::Mul**.

virtual NUMBER::operator CMPLX () const `[pure virtual]`

Оператор приведения к комплексному числу.

Замещается в **CMPLX::operator CMPLX** и **REAL::operator CMPLX**.

virtual void NUMBER::Print () const `[pure virtual]`

Вывод числа на экран.

Замещается в **CMPLX::Print** и **REAL::Print**.

virtual CMPLX NUMBER::Sub (CMPLX *what*) const `[pure virtual]`

Операция деления на комплексное число.

Аргументы метода представлены в таблице 4.

Аргументы:

Таблица 4 – Аргументы метода вычитания

what	делитель

Возвращает:

результат деления

Замещается в **REAL::Sub** и **CMPLX::Sub**.

Класс CMPLX

Класс комплексных чисел.

Является производным от класса **NUMBER**.

```
#include <CMPLX.h>
```

Открытые члены

- **CMPLX** ()
- **CMPLX** (float real, float imag)
- void **Setcompl** (float real, float imag)
- void **Inc** ()
- void **Dec** ()
- **CMPLX Add** (**CMPLX** what) const
- **CMPLX Sub** (**CMPLX** what) const
- **CMPLX Mul** (**CMPLX** what) const
- **CMPLX Cmul** (float what) const
- **CMPLX Div** (**CMPLX** what) const
- void **Print** () const
- **operator CMPLX** () const

Закрытые данные

- float **re**

Вещественная часть

- float **im**

Мнимая часть

Конструкторы

CMPLX::CMPLX ()

Конструктор по умолчанию.

CMPLX::CMPLX (float *real*, float *imag*)

Конструктор, задающий вещественную и мнимую части.

Аргументы конструктора представлены в таблице 5.

Аргументы:

Таблица 5 – Аргументы конструктора (float,float)

real	вещественная часть
imag	мнимая часть

Методы

CMPLX CMPLX::Add (CMPLX *what*) const `[virtual]`

Операция сложения комплексных чисел.

Замещает **NUMBER::Add**.

Аргументы метода представлены в таблице 6.

Аргументы:

Таблица 6 – Аргументы метода сложения

what	второе слагаемое

Возвращает:

сумму комплексных чисел.

CMPLX CMPLX::Cmul (float *what*) const

Операция умножения на вещественное число.

Аргументы метода представлены в таблице 7.

Аргументы:

Таблица 7 – Аргументы метода умножения на вещественное

what	множитель

Возвращает:

результат умножения.

void CMPLX::Dec () const `[virtual]`

Декремент - уменьшение модуля на единицу.

Замещает **NUMBER::Dec**.

CMPLX CMPLX::Div (CMPLX *what*) const `[virtual]`

Операция деления комплексных чисел.

Замещает **NUMBER::Div**.

Аргументы метода представлены в таблице 8.

Аргументы:

Таблица 8 – Аргументы метода деления

what	делитель

Возвращает:

результат деления.

void CMPLX::Inc () `[virtual]`

Инкремент - увеличение модуля на единицу.

Замещает **NUMBER::Inc**.

CMPLX CMPLX::Mul (CMPLX *what*) const `[virtual]`

Операция умножения комплексных чисел.

Замещает **NUMBER::Mul**.

Аргументы метода представлены в таблице 9.

Аргументы:

Таблица 9 – Аргументы метода умножения

what	множитель

Возвращает:

результат умножения.

CMPLX::operator CMPLX () const [virtual]

Оператор приведения к комплексному числу.

Замещает **NUMBER::operator CMPLX**.

void CMPLX::Print () const [virtual]

Вывод числа на экран в формате (real,imag).

Замещает **NUMBER::Print**.

void CMPLX::Setcompl (float *real*, float *imag*)

Установка значения.

Аргументы метода представлены в таблице 10.

Аргументы:

Таблица 10 – Аргументы метода установки значения

real	вещественная часть
imag	мнимая часть

CMPLX CMPLX::Sub (CMPLX *what*) const [virtual]

Операция вычитания комплексных чисел.

Замещает **NUMBER::Sub**.

Аргументы метода представлены в таблице 11.

Аргументы:

Таблица 11 – Аргументы метода вычитания

what	вычитаемое

Класс REAL

Класс "вещественное число".

Является производным от класса **NUMBER**.

```
#include <REAL.h>
```

Открытые члены

- **REAL** ()
- **REAL** (double **value**)
- double **value** ()
- void **setValue** (const double **value**)
- void **Inc** ()

- void **Dec** ()
- **CMPLX Add** (**CMPLX** what) const
- **CMPLX Mul** (**CMPLX** what) const
- **CMPLX Div** (**CMPLX** what) const
- **CMPLX Sub** (**CMPLX** what) const
- void **Print** () const
- **operator CMPLX** () const

Закрытые данные

- double **val**

значение числа

Конструктор(ы)

REAL::REAL ()

Конструктор по умолчанию.

REAL::REAL (double *value*)

Конструктор, задающий значение числа.

Аргументы конструктора представлены в таблице 12.

Аргументы:

Таблица 12 – Аргументы конструктора (double)

value	значение числа

Методы

CMPLX REAL::Add (CMPLX *what*) const `[virtual]`

Операция сложения с комплексным числом.

Аргументы метода представлены в таблице 13.

Аргументы:

Таблица 13 – Аргументы метода сложения

what	слагаемое

Возвращает:

сумму комплексных чисел

Замещает **NUMBER::Add**.

void REAL::Dec () `[virtual]`

Декремент - уменьшение числа на единицу.

Замещает **NUMBER::Dec**.

CMPLX REAL::Div (CMPLX *what*) const `[virtual]`

Операция деления на комплексное число.

Аргументы метода представлены в таблице 14.

Аргументы:

Таблица 14 – Аргументы метода деления

what	делитель

Возвращает:

результат деления

Замещает **NUMBER::Div**.

void REAL::Inc () [virtual]

Инкремент - увеличение числа на единицу.

Замещает **NUMBER::Inc**.

CMPLX REAL::Mul (CMPLX *what*) const [virtual]

Операция умножения на комплексное число.

Аргументы метода представлены в таблице 15.

Аргументы:

Таблица 15 – Аргументы метода умножения

what	множитель

Возвращает:

результат умножения

Замещает **NUMBER::Mul**.

REAL::operator CMPLX () const [virtual]

Оператор приведения к комплексному числу.

Замещает **NUMBER::operator CMPLX**.

void REAL::Print () const [virtual]

Вывод числа на экран.

Замещает **NUMBER::Print**.

void REAL::setValue (const double *value*)

Установка значения числа.

Аргументы метода представлены в таблице 16.

Аргументы:

Таблица 16– Аргументы метода установки значения

value	новое значение числа

CMPLX REAL::Sub (CMPLX *what*) const [virtual]

Операция вычитания комплексного числа.

Аргументы метода представлены в таблице 17.

Аргументы:

Таблица 17 – Аргументы метода вычитания

what	вычитаемое

Возвращает:

разность комплексных чисел

Замещает **NUMBER::Sub**.

double REAL::value ()

Получение значения числа.

Возвращает:

значение числа

Инструкция пользователю

Данная программа производит инкремент и вычитание комплексных и вещественных чисел.

Для работы введите комплексное число, введя вещественную и мнимую части через пробел, затем вещественное число.

После этого программа выведет значения этих чисел, результат их инкремента и разности, сначала комплексного с вещественным, затем наоборот.

Контрольный пример

Ниже на рисунке 1 представлен пример работы программы, работающей с классами чисел с использованием виртуальных функций.

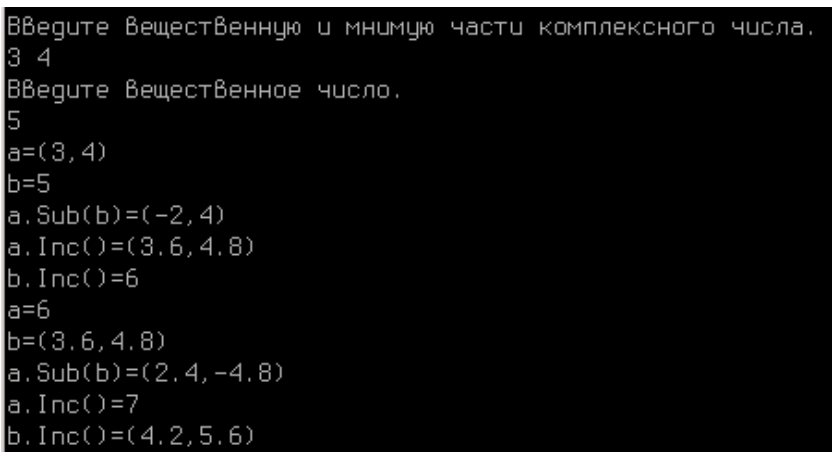

Рисунок 1— Пример работы программы, оперирующей числами с помощью виртуальных функций

Вывод

В данной лабораторной работе я познакомился с наследованием классов и виртуальными функциями в языке программирования C++. Была написана программа, использующая абстрактный базовый класс «число», содержащий чистые виртуальные функции, и производные от него - класс комплексных чисел и класс вещественных чисел, которые по разному реализуют различные методы. Передача параметров в тестовую функцию по ссылке типа абстрактного базового класса позволяет этой функции вызывать методы, реализованные в производных классах; таким образом достигается полиморфизм с помощью виртуальных функций.