Московский авиационный институт
(Национальный исследовательский университет)

Факультет: «Информационные технологии и прикладная математика»
Кафедра: 806 «Вычислительная математика и программирование»
Дисциплина: «Объектно-ориентированное программирование»

Лабораторная работа № 2

Тема: Операторы и литералы C++

Студент: Пермяков Никита Александрович
Группа: 80-208
Преподаватель: Чернышов Л.Н.
Дата:
Оценка:

Москва, 2020

Постановка задачи

Цель: Изучение механизмов перегрузки операторов. Изучение механизмов работы с пользовательскими литералами.

Комплексное число в алгебраической форме представляются парой действ ительных чисел (a, b), где a – действительная часть, b – мнимая часть. Реал изовать класс Complex для работы с комплексными числами.

сложения add, (a, b) + (c, d) = (a + c, b + d);
вычитания sub, (a, b) – (c, d) = (a – c, b – d);
умножения mul, (a, b) × (c, d) = (ac – bd, ad + bc);
деления div, (a, b) / (c, d) = (ac + bd, bc – ad) / (c2 + d2);
сравнение equ, (a, b) = (c, d), если (a = c) и (b = d);
сопряженное число conj, conj(a, b) = (a, –b).
Реализовать операции сравнения модулей.

Операции сложения, вычитания, умножения, деления, сравнения на равенс тво должны быть выполнены в виде перегрузки операторов.
Необходимо реализовать пользовательский литерал для работы с констант ами типа Complex.

Описание программы

Программа получает на вход 2 числа типа double. Преобразует в класс Complex. После вычислений программа выводит результаты операций: +, -, /, *, а также операции сравнения и проверки на сопряженное число, выполненные с помощью перегрузки операторов. Далее программа демонстрирует работу литерала, который приводит строку в нужный вид Complex. Сами функции проделывают данную операцию с составляющими double, и возвращают переменную в виде комплексного числа. Также я сделал пользовательский литерал для работы с константами _REAL и _IMG через вспомогательный класс Part. Перед литералом пишется строка с разделителем «.»

Набор тестов

Пояснение:

1. Вводим две битовые строки
2. Взаимодействуем с текстовым интерфейсом выполняя все операции в случайном порядке.
3. Сравниваем результат битовых операций с выводом ответов в консоль.

Тесты № 1:

5

6

3

4

7

43

87.87

46

76

4

1

17

-2

-667

-453

3

1

65

9

-9

-4

1

8

Тесты № 2:

-5

-6

-3

4

7

43

87.84

46

76

1

1

67

-62

-6667

-4733

3

6265

98

89

-54

1

j

Тесты № 3:

6

2

8

5

4

1987.87

76

46

1

1

67

-62	1	-54
-6667	6865	1
-4753	98	p
3	-59	

Результаты выполнения тестов.

```
Enter the real and img of complex no.
Real : 5
Img : 6
Enter the real and img of complex no.
Real : 3
Img : 4


        ---Main Menu--
        1.Addition
        2.Subtraction
        3.Multiplication
        4.Division
        5.Equal
        6.Conjugate
        7.Show number with
          Users literals
          in Complex define
        8.Exit
       ---Enter your choice--> 7
```

```
(4.74) + (5.46)i


        ---Press 1 to continue--> ^C
root@dell:/mnt/c/Users/permi/source/repos/university/oop/2_operator_liter# g++ main.cpp
root@dell:/mnt/c/Users/permi/source/repos/university/oop/2_operator_liter# ./a.out
Enter the real and img of complex no.
Real : 43
Img : 87.87
Enter the real and img of complex no.
Real :
46
Img : 76


        ---Main Menu--
        1.Addition
        2.Subtraction
        3.Multiplication
        4.Division
        5.Equal
        6.Conjugate
        7.Show number with
          Users literals
          in Complex define
        8.Exit
       ---Enter your choice--> 4
```

(43) + (87.87)i / (46) + (76)i = (1.09682) + (0.0980765)i

---Press 1 to continue-->

1

Enter the real and img of complex no.

Real : 17

Img : -2

Enter the real and img of complex no.

Real : -667

Img : -453

---Main Menu--

1.Addition

2.Subtraction

3.Multiplication

4.Division

5.Equal

6.Conjugate

7.Show number with

Users literals

in Complex define

8.Exit

---Enter your choice--> 3

(17) + (-2)i * (-667) + (-453)i = (-12245) + (-6367)i

---Press 1 to continue--> 1

Enter the real and img of complex no.

Real : 65

Img : 9

Enter the real and img of complex no.

Real : -9

Img : -4

---Main Menu--

1.Addition

2.Subtraction

3.Multiplication

4.Division

5.Equal

6.Conjugate

7.Show number with

Users literals

in Complex define

8.Exit

---Enter your choice--> 1

(65) + (9)i + (-9) + (-4)i = (56) + (5)i

---Press 1 to continue--> 8

Листинг программы

```
/*
Комплексное число в алгебраической форме представляются парой действ
ительных чисел (a, b), где a – действительная часть, b – мнимая част
ь. Реализовать класс Complex для работы с комплексными числами. Обяз
ательно должны быть присутствовать операции
сложения add, (a, b) + (c, d) = (a + c, b + d);
вычитания sub, (a, b) – (c, d) = (a – c, b – d);
умножения mul, (a, b) × (c, d) = (ac – bd, ad + bc);
деления div, (a, b) / (c, d) = (ac + bd, bc – ad) / (c2 + d2);
сравнение equ, (a, b) = (c, d), если (a = c) и (b = d);
сопряженное число conj, conj(a, b) = (a, –b).
Реализовать операции сравнения модулей.

Операции сложения, вычитания, умножения, деления, сравнения на равен
ство должны быть выполнены в виде перегрузки операторов.
Необходимо реализовать пользовательский литерал для работы с констан
тами типа Complex.
*/

#include<iostream>
#include<cmath>


using namespace std;

class Complex {
public:
  Complex() {
    this->real = 0;
    this->img = 0;
  }

  void setRealImg(double first, double second) {
    this->real = first;
    this->img = second;
  }
  void getData();
  void display();

  Complex operator+ (Complex c1);
```

```
  Complex operator- (Complex c1);
  Complex operator* (Complex c1);
  Complex operator/ (Complex c1);
  bool operator== (Complex c1);
  bool operator^ (Complex c1);
  float abs();

private:
  float real;
  float img;
};


void Complex::display() {
 cout << "(" << this->real << ")" << " + " << "(" << this->img << "
)" << "i";
}

void Complex::getData() {
  cout << "Enter the real and img of complex no.\n";
  cout << "Real : ";
  cin >> this->real;
  cout << "Img : ";
  cin >> this->img;
}

Complex Complex::operator/ (Complex c1) {
  Complex div;
  div.real = ( (this->real * c1.real) + (this->img * c1.img) ) / ( p
ow(c1.real, 2) + pow(c1.img, 2) );
  div.img = ( (this->img * c1.real) - (this->real * c1.img) ) / ( po
w(c1.real, 2) + pow(c1.img, 2) );
  return div;
}

Complex Complex::operator* (Complex c1) {
  Complex mul;
  mul.real = ( (this->real * c1.real) - (this->img * c1.img) );
  mul.img = ( (this->real * c1.img) + (this->img * c1.real) );
  return mul;
}
```

```
Complex Complex::operator- (Complex c1) {
  Complex sub;
  sub.real = this->real - c1.real;
  sub.img = this->img - c1.img;
  return sub;
}

Complex Complex::operator+ (Complex c1) {
  Complex add;
  add.real = this->real + c1.real;
  add.img = this->img + c1.img;
  return add;
}

bool Complex::operator== (Complex c1) {
  if (this->real == c1.real && this->img == c1.img){
    return true;
  }
  return false;
}

bool Complex::operator^ (Complex c1) {
  if (this->real == c1.real && this->img == -1 * c1.img) {
    return true;
  }
  return false;
}

float Complex::abs() {
  float max_part = max(this->real, this->img);
  float min_part = min(this->real, this->img);
  return max_part * sqrt(1 + min_part * min_part / max_part / max_pa
rt);
}

class Part {
public:
  Part() {
    this->part = 0;
  }
  Part(long double part){
    this->part = part;
```

```
  }
  double part;
};

Part operator ""_REAL(long double part) {
  Part lit(part);
  return lit;
}

Part operator ""_IMG(long double part) {
  Part lit(part);
  return lit;
}

int main() {
  Complex a, b, c;
  Complex example;
  Part part_real, part_img;

  int opt, opt_exit = 1;

  while(opt_exit == 1) {
    a.getData();
    b.getData();

    cout << "\n\t ---Main Menu--";
    cout << "\n\t 1.Addition";
    cout << "\n\t 2.Subtraction";
    cout << "\n\t 3.Multiplication";
    cout << "\n\t 4.Division";
    cout << "\n\t 5.Equal";
    cout << "\n\t 6.Conjugate";
    cout << "\n\t 7.Show number with\n\t   Users literals\n\t   in C
omplex define";
    cout << "\n\t 8.Exit";
    cout << "\n\t---Enter your choice--> ";

    cin >> opt;

    switch(opt) {
    case 1:
      c = a + b;
```

```
  cout << "\n\n";

  a.display();
  cout << " + ";
  b.display();
  cout << " = ";
  c.display();
  break;

case 2:
  c = a - b;
  cout << "\n\n";
  a.display();
  cout << " - ";
  b.display();
  cout << " = ";
  c.display();
  break;

case 3:
  c = a * b;
  cout << "\n\n";
  a.display();
  cout << " * ";
  b.display();
  cout << " = ";
  c.display();
  break;

case 4:
  c = a / b;
  cout << "\n\n";
  a.display();
  cout << " / ";
  b.display();
  cout << " = ";
  c.display();
  break;

case 5:
  cout << "\n\t   Numbers are absolutes? (1 is yes) --> ";
  cin >> opt;
```

```
  cout << "\n\n";
  if (opt == 1){
    cout << a.abs();
    if (a.abs() == b.abs())
      cout << " == ";
    else
      cout << " != ";
    cout << b.abs();
  } else {
    a.display();
    if (a == b)
      cout << " == ";
    else
      cout << " != ";
    b.display();
  }
  break;

case 6:
  cout << "\n\n";
  a.display();
  if (a ^ b)
    cout << "  =  ";
  else
    cout << " =\\= ";
  b.display();
  break;

case 7:
  part_real = 4.74_REAL;
  part_img = 5.46_IMG;
  example.setRealImg(part_real.part, part_img.part);
  example.display();
  break;

case 8:
  return 0;

default:
  cout << "\n\n\t---Invalid choice..... try again\n";
  break;
}
```

```
    cout << "\n\n\t---Press 1 to continue--> ";
    cin >> opt_exit;
  }
 return 0;
}
```

Вывод

В работе были освоенны навыки перегрузки операторов для класса комплексных чисел. Освоенны литералы — возможность языка определять пользовательские типы данных. Изученна возможность нестатического использования литералов. Перегрузка операций предполагает введение двух взаимосвязанных особенностей: возможности объявлять в одной области видимости несколько процедур или функций с одинаковыми именами и возможности описывать собственные реализации бинарных операторов (то есть знаков операций, обычно записываемых в инфиксной нотации, между операндами). Перегрузить операторы можно как через дружеские функции, так и с помощью методов класса. Но стоит помнить, что некоторые операторы перегрузить нельзя:
“::” “.” “.*” “sizeof”, “typeid”.

СПИСОК ЛИТЕРАТУРЫ

1. Перегрузка операторов C++ [Электронный ресурс]. URL: https://metanit.com/cpp/tutorial/5.14.php

(дата обращения: 29.09.2019).

2. Битовые операции C++ [Электронный ресурс]. URL: http://www.c-cpp.ru/books/bitovye-operatory

(дата обращения: 29.09.2019).