**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Объектно-ориентированное программирование»

**Лабораторная работа № 1**

Тема: Простые классы на языке С++

Студент: Лейченко Кира

Группа: М8О-201Б

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2019

1. Постановка задачи

Вариант 5:

Создать класс Module для работы с целыми числами по модулю N. В классе должно быть два поля: число и N. Реализовать все арифметические операции. Реализовать операции сравнения.

1. Описание программы.

Программа хранит число M и его модуль N, изначально M=0, N=1. Значение M и N можно задать вручную, если пользователь ввел N<=0, вызывается исключение (“Wrong modul”). Функция нахождения остатка от деления M на N выводит число (цифра 1). Допустимые операции с объектами класса Module(цифра 2)(+,-,\*,/-подпункты меню ‘2’ [1-4] <,><=,>=,==,!= -подпункты меню ‘2’ [5-10]. Если происходит деление на 0 вызывается исключение (“Division by zero”). Пункт меню 3 вызывает функцию Writing\_values, в которой пользователь вводит новые данные

Ссылка на **github:** <https://github.com/Leyka17/Labs_OOP/tree/master/01_Lab>

1. Набор testcases

|  |  |  |
| --- | --- | --- |
| № | Описание | Ввод |
| 1 | Демонстрация работы программы на простых примерах | 7  3  4  1  2  1  2  2  2  3  2  4  2  5  2  6  2  7  2  8  2  9  2  10  3  4  -7  8  2  2  2  2  4  0 |
| 2 | Демонстрация работы программы на более сложных примерах: | 85  -7898965  124  2  8  2  9  2  4  1  0 |
| 3 | Тесты вызывающие исключения | 0  5  4  0  2  4 |

1. Результаты выполнения тестов.

test\_1.txt:

This is menu for working with class integers numbers by modul N

Write modul N

N=7

Write value for first number

3

Write value for second number

4

Menu

1.Find modulo value

2.Operation with class's ojects

3.Write new values

0.Exit

Write comand

1

Modulo value: 3

Modulo value: 4

Menu

1.Find modulo value

2.Operation with class's ojects

3.Write new values

0.Exit

Write comand

2

Operation with class's objects

1.Operation +

2.Operation -

3.Operation \*

4.Operation /

5.Operation >

6.Operation <

7.Operation >=

8.Operation <=

9.Operation ==

10.Operation !=

0.exit

1

adding

A+B= Value: 7 Module: 7 Modulo value: 0

Menu

1.Find modulo value

2.Operation with class's ojects

3.Write new values

0.Exit

Write comand

2

Operation with class's objects

1.Operation +

2.Operation -

3.Operation \*

4.Operation /

5.Operation >

6.Operation <

7.Operation >=

8.Operation <=

9.Operation ==

10.Operation !=

0.exit

2

subtraction

A-B= Value: -1 Module: 7 Modulo value: 1

Menu

1.Find modulo value

2.Operation with class's ojects

3.Write new values

0.Exit

Write comand

2

Operation with class's objects

1.Operation +

2.Operation -

3.Operation \*

4.Operation /

5.Operation >

6.Operation <

7.Operation >=

8.Operation <=

9.Operation ==

10.Operation !=

0.exit

3

multing

A\*B= Value: 12 Module: 7 Modulo value: 5

Menu

1.Find modulo value

2.Operation with class's ojects

3.Write new values

0.Exit

Write comand

2

Operation with class's objects

1.Operation +

2.Operation -

3.Operation \*

4.Operation /

5.Operation >

6.Operation <

7.Operation >=

8.Operation <=

9.Operation ==

10.Operation !=

0.exit

4

division

A/B= Value: 0 Module: 7 Modulo value: 0

Menu

1.Find modulo value

2.Operation with class's ojects

3.Write new values

0.Exit

Write comand

2

Operation with class's objects

1.Operation +

2.Operation -

3.Operation \*

4.Operation /

5.Operation >

6.Operation <

7.Operation >=

8.Operation <=

9.Operation ==

10.Operation !=

0.exit

5

>

Result A>B: 0

Menu

1.Find modulo value

2.Operation with class's ojects

3.Write new values

0.Exit

Write comand

2

Operation with class's objects

1.Operation +

2.Operation -

3.Operation \*

4.Operation /

5.Operation >

6.Operation <

7.Operation >=

8.Operation <=

9.Operation ==

10.Operation !=

0.exit

6

<

Result A<B: 1

Menu

1.Find modulo value

2.Operation with class's ojects

3.Write new values

0.Exit

Write comand

2

Operation with class's objects

1.Operation +

2.Operation -

3.Operation \*

4.Operation /

5.Operation >

6.Operation <

7.Operation >=

8.Operation <=

9.Operation ==

10.Operation !=

0.exit

7

>=l

Result A>=B: 0

Menu

1.Find modulo value

2.Operation with class's ojects

3.Write new values

0.Exit

Write comand

2

Operation with class's objects

1.Operation +

2.Operation -

3.Operation \*

4.Operation /

5.Operation >

6.Operation <

7.Operation >=

8.Operation <=

9.Operation ==

10.Operation !=

0.exit

8

<=

Result A<=B: 1

Menu

1.Find modulo value

2.Operation with class's ojects

3.Write new values

0.Exit

Write comand

2

Operation with class's objects

1.Operation +

2.Operation -

3.Operation \*

4.Operation /

5.Operation >

6.Operation <

7.Operation >=

8.Operation <=

9.Operation ==

10.Operation !=

0.exit

9

==

Result A==B: 0

Menu

1.Find modulo value

2.Operation with class's ojects

3.Write new values

0.Exit

Write comand

2

Operation with class's objects

1.Operation +

2.Operation -

3.Operation \*

4.Operation /

5.Operation >

6.Operation <

7.Operation >=

8.Operation <=

9.Operation ==

10.Operation !=

0.exit

10

!=

Result A!=B: 1

Menu

1.Find modulo value

2.Operation with class's ojects

3.Write new values

0.Exit

Write comand

3

Write modul N

N=4

Write value for first number

-7

Write value for second number

8

Menu

1.Find modulo value

2.Operation with class's ojects

3.Write new values

0.Exit

Write comand

2

Operation with class's objects

1.Operation +

2.Operation -

3.Operation \*

4.Operation /

5.Operation >

6.Operation <

7.Operation >=

8.Operation <=

9.Operation ==

10.Operation !=

0.exit

2

subtraction

A-B= Value: -15 Module: 4 Modulo value: 3

Menu

1.Find modulo value

2.Operation with class's ojects

3.Write new values

0.Exit

Write comand

2

Operation with class's objects

1.Operation +

2.Operation -

3.Operation \*

4.Operation /

5.Operation >

6.Operation <

7.Operation >=

8.Operation <=

9.Operation ==

10.Operation !=

0.exit

2

subtraction

A-B= Value: -15 Module: 4 Modulo value: 3

Menu

1.Find modulo value

2.Operation with class's ojects

3.Write new values

0.Exit

Write comand

4

Menu

1.Find modulo value

2.Operation with class's ojects

3.Write new values

0.Exit

Write comand

0

Thank you for using

**Test\_2.txt**

This is menu for working with class integers numbers by modul N

Write modul N

N=85

Write value for first number

-7898965

Write value for second number

124

Menu

1.Find modulo value

2.Operation with class's ojects

3.Write new values

0.Exit

Write comand

2

Operation with class's objects

1.Operation +

2.Operation -

3.Operation \*

4.Operation /

5.Operation >

6.Operation <

7.Operation >=

8.Operation <=

9.Operation ==

10.Operation !=

0.exit

8

<=

Result A<=B: 1

Menu

1.Find modulo value

2.Operation with class's ojects

3.Write new values

0.Exit

Write comand

2

Operation with class's objects

1.Operation +

2.Operation -

3.Operation \*

4.Operation /

5.Operation >

6.Operation <

7.Operation >=

8.Operation <=

9.Operation ==

10.Operation !=

0.exit

9

==

Result A==B: 0

Menu

1.Find modulo value

2.Operation with class's ojects

3.Write new values

0.Exit

Write comand

2

Operation with class's objects

1.Operation +

2.Operation -

3.Operation \*

4.Operation /

5.Operation >

6.Operation <

7.Operation >=

8.Operation <=

9.Operation ==

10.Operation !=

0.exit

4

division

A/B= Value: -63701 Module: 85 Modulo value: 36

Menu

1.Find modulo value

2.Operation with class's ojects

3.Write new values

0.Exit

Write comand

1

Modulo value: 0

Modulo value: 39

Menu

1.Find modulo value

2.Operation with class's ojects

3.Write new values

0.Exit

Write comand

0

Thank you for using

**Test\_3.txt**

This is menu for working with class integers numbers by modul N

Write modul N

N=0

You write incorrect modul: 0

**Test\_4.txt**

This is menu for working with class integers numbers by modul N

Write modul N

N=5

Write value for first number

4

Write value for second number

0

Menu

1.Find modulo value

2.Operation with class's ojects

3.Write new values

0.Exit

Write comand

2

Operation with class's objects

1.Operation +

2.Operation -

3.Operation \*

4.Operation /

5.Operation >

6.Operation <

7.Operation >=

8.Operation <=

9.Operation ==

10.Operation !=

0.exit

4

division

Division by zero, becouse leftOperand==0

1. Листинг программы.

Module.h

#include<iostream>

class Module

{

private:

int \_num;

int \_modul;

public:

//exeptions

class Wrongmodul

{

public:int \_modul;

Wrongmodul(int modul) :\_modul{ modul } {}

};

class DivisionbyZero {};

class DiffrentModuls

{

public: int \_modul1;

int \_modul2;

DiffrentModuls(int modul1, int modul2) :\_modul1{ modul1 }, \_modul2{ modul2 } {}

};

//ctors

explicit Module(int \_num = 0, int \_modul = 1);//нет неявного присвоения нельзя module=8,7

//copy ctor

Module(const Module&other);

//del ctor

~Module();

//Getter and setter

int GetNum()const;

int GetModul()const;

void SetNum(const int&value);

void SetModul(const int &value);

//functions

int FindModul()const;

int FindModulForInt(int val, int modul)const;

//operators

Module&operator=(Module&&other);//for move itemsfor copy items

Module& operator = (const Module& other);//for copy items

};

Module operator+(const Module&leftOperand, const Module &rightOperand);

Module operator+(const Module& leftOperand, const int& rightOperand);

Module operator+(const int& leftOperand, const Module& rightOperand);

Module operator-(const Module&leftOperand, const Module &rightOperand);

Module operator-(const Module& leftOperand, const int& rightOperand);

Module operator-(const int& leftOperand, const Module& rightOperand);

Module operator/(const Module&leftOperand, const Module &rightOperand);

Module operator/(const Module& leftOperand, const int& rightOperand);

Module operator/(const int& leftOperand, const Module& rightOperand);

Module operator\*(const Module&leftOperand, const Module &rightOperand);

Module operator\*(const Module& leftOperand, const int& rightOperand);

Module operator\*(const int& leftOperand, const Module& rightOperand);

bool operator==(const Module& leftOperand, const Module&rightOperand);

bool operator==(const Module& leftOperand,const int& rightOperand);

bool operator==(const int& leftOperand, const Module&rightOperand);

bool operator!=(const Module& leftOperand, const Module&rightOperand);

bool operator!=(const Module& leftOperand, const int& rightOperand);

bool operator!=(const int& leftOperand, const Module&rightOperand);

bool operator>(const Module& leftOperand, const Module&rightOperand);

bool operator>(const Module& leftOperand, const int& rightOperand);

bool operator>(const int& leftOperand, const Module&rightOperand);

bool operator<(const Module& leftOperand, const Module&rightOperand);

bool operator<(const Module& leftOperand, const int& rightOperand);

bool operator<(const int& leftOperand, const Module&rightOperand);

bool operator>=(const Module& leftOperand, const Module&rightOperand);

bool operator>=(const Module& leftOperand, const int& rightOperand);

bool operator>=(const int& leftOperand, const Module&rightOperand);

bool operator<=(const Module& leftOperand, const Module&rightOperand);

bool operator<=(const Module& leftOperand, const int& rightOperand);

bool operator<=(const int& leftOperand, const Module&rightOperand);

std::ostream& operator << (std::ostream& stream, const Module& toOutput);

std::istream& operator >> (std::istream& stream, Module& toInput);

Module.cpp

#include"Module.h"

#include <iostream>

#include<algorithm>

#include<cmath>

//ctors

Module::Module(int num, int modul) :\_num {num}

{

if (modul <=0)

{

throw Wrongmodul(modul);//raise exeption

}

\_modul = modul;

}

//copy ctor

Module::Module(const Module&other)

{

\*this = other;

}

//del ctor

Module::~Module()

{

}

//Getter and Setter

int Module::GetModul()const

{

return \_modul;

}

int Module::GetNum()const

{

return \_num;

}

void Module::SetModul(const int &modul)

{

if (modul<= 0)

{

throw Wrongmodul(modul);

}

\_modul =modul;

}

void Module::SetNum(const int &num)

{

\_num = num;

}

//functions

int Module::FindModul()const

{

return abs( \_num - (\_num / \_modul)\*\_modul);

}

int Module::FindModulForInt(int num, int modul)const

{

if (modul <= 0)

{

throw Wrongmodul(modul);

}

return abs( num - (num / modul)\*modul);

}

//operators which belong to class

Module& Module::operator=(const Module &other)

{

\_num = other.GetNum();

\_modul = other.GetModul();

return \*this;

}

//Принимаем не константную ссылку, так как объект other должен изменить значение

Module& Module::operator=( Module &&other)

{

std::swap(\_num, other.\_num);

std::swap(\_modul, other.\_modul);

return \*this;

}

Module operator+(const Module& leftOperand, const Module& rightOperand)

{

if (leftOperand.GetModul() == rightOperand.GetModul())

{

Module temporary;

temporary.SetNum(leftOperand.GetNum() + rightOperand.GetNum());

temporary.SetModul(leftOperand.GetModul());

return temporary;

}

throw Module::DiffrentModuls(leftOperand.GetModul(),rightOperand.GetModul());

}

Module operator+(const int& leftOperand, const Module& rightOperand)

{

Module temporary;

temporary.SetNum(leftOperand + rightOperand.GetNum());

temporary.SetModul(rightOperand.GetModul());

return temporary;

}

Module operator+(const Module& leftOperand, const int& rightOperand)

{

Module temporary;

temporary.SetNum(leftOperand.GetNum()+ rightOperand);

temporary.SetModul( leftOperand.GetModul());

return temporary;

}

Module operator-(const Module& leftOperand, const Module& rightOperand)

{

if (leftOperand.GetModul() == rightOperand.GetModul())

{

Module temporary;

temporary.SetNum( leftOperand.GetNum() - rightOperand.GetNum());

temporary.SetModul (leftOperand.GetModul());

return temporary;

}

throw Module::DiffrentModuls(leftOperand.GetModul(), rightOperand.GetModul());

}

Module operator-(const Module& leftOperand, const int& rightOperand)

{

Module temporary;

temporary.SetNum ( leftOperand.GetNum() - rightOperand);

temporary.SetModul(leftOperand.GetModul());

return temporary;

}

Module operator-(const int& leftOperand, const Module& rightOperand)

{

Module temporary;

temporary.SetNum (leftOperand - rightOperand.GetNum());

temporary.SetModul (rightOperand.GetModul());

return temporary;

}

Module operator/(const Module& leftOperand, const Module& rightOperand)

{

if (rightOperand.GetNum())

{

if (leftOperand.GetModul() == rightOperand.GetModul())

{

Module temporary;

temporary.SetNum(leftOperand.GetNum() / rightOperand.GetNum());

temporary.SetModul(leftOperand.GetModul());

return temporary;

}

throw Module::DiffrentModuls(leftOperand.GetModul(), rightOperand.GetModul());

}

throw Module::DivisionbyZero();

}

Module operator/(const Module& leftOperand, const int& rightOperand)

{

if(rightOperand)

{

Module temporary;

temporary.SetNum(leftOperand.GetNum() /rightOperand);

temporary.SetModul ( leftOperand.GetModul());

return temporary;

}

throw Module::DivisionbyZero();

}

Module operator/(const int& leftOperand, const Module& rightOperand)

{

if(rightOperand.GetNum())

{

Module temporary;

temporary.SetNum(leftOperand / rightOperand.GetNum());

temporary.SetModul(rightOperand.GetModul());

return temporary;

}

throw Module::DivisionbyZero();

}

Module operator\*(const Module& leftOperand, const Module& rightOperand)

{

if (leftOperand.GetModul() == rightOperand.GetModul())

{

Module temporary;

temporary.SetNum( leftOperand.GetNum() \* rightOperand.GetNum());

temporary.SetModul(leftOperand.GetModul());

return temporary;

}

throw Module::DiffrentModuls(leftOperand.GetModul(), rightOperand.GetModul());

}

Module operator\*(const Module& leftOperand, const int& rightOperand)

{

Module temporary;

temporary.SetNum(leftOperand.GetNum() \* rightOperand);

temporary.SetModul( leftOperand.GetModul());

return temporary;

}

Module operator\*(const int& leftOperand, const Module& rightOperand)

{

Module temporary;

temporary.SetNum( leftOperand \* rightOperand.GetNum());

temporary.SetModul( rightOperand.GetModul());

return temporary;

}

bool operator==(const Module& leftOperand, const Module& rightOperand)

{

if((leftOperand.GetModul()==rightOperand.GetModul())&&(leftOperand.FindModul()==rightOperand.FindModul()))

return true;

return false;

}

bool operator==(const int& leftOperand, const Module& rightOperand)

{

if(rightOperand.FindModulForInt(leftOperand,rightOperand.GetModul())==rightOperand.FindModul())

return true;

return false;

}

bool operator==(const Module& leftOperand, const int& rightOperand)

{

if (leftOperand.FindModul() == leftOperand.FindModulForInt(rightOperand,leftOperand.GetModul()))

return true;

return false;

}

bool operator!=(const Module& leftOperand, const Module& rightOperand)

{

return !(leftOperand == rightOperand);

}

bool operator!=(const int& leftOperand, const Module& rightOperand)

{

return !(leftOperand == rightOperand);

}

bool operator!=(const Module& leftOperand, const int& rightOperand)

{

return !(leftOperand == rightOperand);

}

bool operator>(const Module& leftOperand, const Module&rightOperand)

{

if (leftOperand.GetModul() == rightOperand.GetModul())

{

if (leftOperand.FindModul() > rightOperand.FindModul())

return true;

return false;

}

throw Module::DiffrentModuls(leftOperand.GetModul(), rightOperand.GetModul());

}

bool operator>(const Module& leftOperand, const int& rightOperand)

{

if (leftOperand.FindModul() > leftOperand.FindModulForInt(rightOperand,leftOperand.GetModul()))

return true;

return false;

}

bool operator>(const int& leftOperand, const Module&rightOperand)

{

if ( rightOperand.FindModulForInt(leftOperand, rightOperand.GetModul()) > rightOperand.FindModul())

return true;

return false;

}

bool operator<(const Module& leftOperand, const Module&rightOperand)

{

if (leftOperand.GetModul() == rightOperand.GetModul())

{

if (leftOperand.FindModul() < rightOperand.FindModul())

return true;

return false;

}

throw Module::DiffrentModuls(leftOperand.GetModul(), rightOperand.GetModul());

}

bool operator<(const Module& leftOperand, const int& rightOperand)

{

if (leftOperand.FindModul() < leftOperand.FindModulForInt(rightOperand,leftOperand.GetModul()))

return true;

return false;

}

bool operator<(const int& leftOperand, const Module&rightOperand)

{

if ( rightOperand.FindModulForInt(leftOperand,rightOperand.GetModul())<rightOperand.FindModul())

return true;

return false;

}

bool operator>=(const Module& leftOperand, const Module&rightOperand)

{

if ((leftOperand == rightOperand) || (leftOperand > rightOperand))

return true;

return false;

}

bool operator>=(const Module& leftOperand, const int& rightOperand)

{

if ((leftOperand == rightOperand) || (leftOperand > rightOperand))

return true;

return false;

}

bool operator>=(const int& leftOperand, const Module&rightOperand)

{

if ((leftOperand == rightOperand) || (leftOperand > rightOperand))

return true;

return false;

}

bool operator<=(const Module& leftOperand, const Module&rightOperand)

{

if ((leftOperand == rightOperand) || (leftOperand < rightOperand))

return true;

return false;

}

bool operator<=(const Module& leftOperand, const int& rightOperand)

{

if ((leftOperand == rightOperand) || (leftOperand < rightOperand))

return true;

return false;

}

bool operator<=(const int& leftOperand, const Module&rightOperand)

{

if ((leftOperand == rightOperand) || (leftOperand < rightOperand))

return true;

return false;

}

std::ostream& operator << (std::ostream& stream, const Module& toOutput)

{

return stream << " Value: " << toOutput.GetNum() << ' ' << "Module: " << toOutput.GetModul() <<' ' <<

" Modulo value: "<<toOutput.FindModul()<<std::endl;

}

std::istream& operator >> (std::istream& stream, Module& toInput)

{

int num;

int modul;

stream >> num >> modul;

toInput.SetNum(num);

toInput.SetModul(modul);

return stream;

}

1\_lab\_OOP.cpp

/\*

♥Лейченко Кира Андреевна M8О-201

♥Вариант 5

♥Создать класс Module для работы с целыми числами по модулю N. В классе должно быть два поля: число и N. Реализовать все арифметические операции. Реализовать операции сравнения.

\*/

#include"Module.h"

#include<iostream>

void Writing\_values(Module&left, Module&right)

{

int modul = 1, value = 0;

std::cout << "Write modul N" << std::endl << "N=";

std::cin >> modul;

right.SetModul(modul);

left.SetModul(modul);

std::cout << "Write value for first number" << std::endl;

std::cin >> value;

left.SetNum(value);

std::cout << "Write value for second number" << std::endl;

std::cin >> value;

right.SetNum(value);

return;

}

int main()

{

try

{

int value = 0;

int modul = 1;

int user\_comand;

bool enterence = true;

Module Input;

Module left, right;

std::cout << "This is menu for working with class integers numbers by modul N" << std::endl;

Writing\_values(left, right);

while (enterence)

{

std::cout << "Menu" << std::endl;

std::cout <<

"1.Find modulo value" << std::endl <<

"2.Operation with class's ojects" << std::endl <<

"3.Write new values" << std::endl<<

"0.Exit" << std::endl;

std::cout << "Write comand" << std::endl;

std::cin >> user\_comand;

switch (user\_comand)

{

case 1:

std::cout << "Modulo value: " << left.FindModul() << std::endl;

std::cout << "Modulo value: " << right.FindModul() << std::endl;

break;

case 2:

{

std::cout << "Operation with class's objects" << std::endl <<

"1.Operation +" << std::endl <<

"2.Operation -" << std::endl <<

"3.Operation \*" << std::endl <<

"4.Operation /" << std::endl <<

"5.Operation >" << std::endl <<

"6.Operation <" << std::endl <<

"7.Operation >=" << std::endl <<

"8.Operation <=" << std::endl <<

"9.Operation ==" << std::endl <<

"10.Operation !=" << std::endl <<

"0.exit" << std::endl;

std::cin >> user\_comand;

switch (user\_comand)

{

case 1:

std::cout << " adding" << std::endl;

std::cout << "A+B= " << (left + right) << std::endl;

break;

case 2:

std::cout << "subtraction" << std::endl;

std::cout << "A-B= " << (left - right) << std::endl;

break;

case 3:

std::cout << "multing" << std::endl;

std::cout << "A\*B= " << (left \* right) << std::endl;

break;

case 4:

std::cout << "division" << std::endl;

std::cout << "A/B= " << (left / right) << std::endl;

break;

case 5:

std::cout << ">" << std::endl;

std::cout << " Result A>B: " << (left > right) << std::endl;

break;

case 6:

std::cout << "<" << std::endl;

std::cout << " Result A<B: " << (left < right) << std::endl;

break;

case 7:

std::cout <<">=l" << std::endl;

std::cout << " Result A>=B: " << (left >= right) << std::endl;

break;

case 8:

std::cout << "<=" << std::endl;

std::cout << " Result A<=B: " << (left <= right) << std::endl;

break;

case 9:

std::cout << "==" << std::endl;

std::cout << " Result A==B: " << (left == right) << std::endl;

break;

case 10:

std::cout << "!=" << std::endl;

std::cout << " Result A!=B: " << (left != right) << std::endl;

break;

}

break;

}

case 3:Writing\_values(left, right); break;

case 0:enterence = false; break;

}

}

std::cout << "Thank you for using" << std::endl;

}

catch(Module::Wrongmodul modul)

{

std::cout << "You write incorrect modul: " << modul.\_modul<<std::endl;

}

catch (Module::DiffrentModuls modul)

{

std::cout << "I can't do it,diffrent moduls:" << std::endl << "Modul\_1: " << modul.\_modul1 << std::endl << "Modul\_2: " << modul.\_modul2<<std::endl;

}

catch (Module::DivisionbyZero)

{

std::cout << "Division by zero, becouse leftOperand==0" << std::endl;

}

system("pause");

return 0;

}

1. Вывод

Реализовал при помощи классов программу на языке С++ по работе с числами и их модулями . Используя перегрузку операторов сделал возможным применение к объектам класса операторов сложения, вычитания и пр. в соответствии с заданием. Корректность работы программы проверил на двух тестовых файлах.

7.Список литературы

1. Шилдт, Герберт. С++: базовый курс, 3-е изд. : Пер. с англ. - М. : ООО “И.Д. Вильямс”, 2018. - 624 с. : ил. - Парал. тит. англ.
2. Справочник по языку С++ [Электронный ресурс]. URL: <http://www.cplusplus.com/reference/deque/> (дата обращения: 14.09.2019).