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 C++

Студент: Суханов Егор Алексеевич
Группа: 80-206
Преподаватель: Чернышов Л.Н.
Дата:
Оценка:

Москва, 2020

Содержание

Код программы

Для удобства использования и редактирования исходный код программы состоит из 3-х файлов:

Modulo.hpp – Заголовочный файл, содержащий объявление класса Modulo.

```
#include "iostream"
#pragma once

/*
  Класс для работы с числами по модулю.
  Перегружает арифметические операции +,-,*,/.
  А так же операции сравнения >,>=,<,<=,==.
  Создает ошибку "diff modulo",
  если операнды имеют разные значения mod.
  Создает ошибку "zero modulo",
  если попытаться задать модуль равный нулю.
*/
class Modulo
{
public:
  Modulo();
  Modulo(int number, int mod);

  /* Получить поле Number */
  int GetNumber();
  /* Получить поле Mod */
  int GetMod();

  Modulo operator+(const Modulo &rhs) const;
  Modulo operator-(const Modulo &rhs) const;
  Modulo operator*(const Modulo &rhs) const;
  Modulo operator/(const Modulo &rhs) const;

  bool operator>(const Modulo &rhs) const;
  bool operator>=(const Modulo &rhs) const;
  bool operator<(const Modulo &rhs) const;
  bool operator<=(const Modulo &rhs) const;
  bool operator==(const Modulo &rhs) const;

  friend std::ostream &operator<<(std::ostream &out, const Modulo &mo
dulo);
  friend std::istream &operator>>(std::istream &in, Modulo &modulo);

private:
  // Проверяет поле mod у this и b на тождество. Если они разные, соз
дает ошибку "diff modulo"
  void validateMod(const Modulo &b) const
```

```
  {
    if (this->mod != b.mod)
      throw "diff modulo";
    if (this->mod == 0)
      throw "zero modulo";
  }

  int number;
  int mod; // Модуль по которому выполняются операции
};
```

Modulo.cpp – файл с определением класса Modulo.

```
#include "modulo.hpp"

Modulo::Modulo()
{
  this->number = 0;
  this->mod = 1;
}
Modulo::Modulo(int number, int mod)
{
  if (mod == 0)
    throw "zero modulo";
  this->number = number % mod;
  this->mod = mod;
}

int Modulo::GetNumber()
{
  return this->number;
}

int Modulo::GetMod()
{
  return this->mod;
}

Modulo Modulo::operator+(const Modulo &rhs) const
{
  validateMod(rhs);
  return {(this->number + rhs.number) % this->mod, mod};
}
Modulo Modulo::operator-(const Modulo &rhs) const
{
  validateMod(rhs);
  return {(this->number - rhs.number) % mod, this->mod};
}
```

```
Modulo Modulo::operator*(const Modulo &rhs) const
{
  validateMod(rhs);
  return {(this->number * rhs.number) % this->mod, this->mod};
}
Modulo Modulo::operator/(const Modulo &rhs) const
{
  validateMod(rhs);
  return {this->number / rhs.number, this->mod};
}

bool Modulo::operator>(const Modulo &rhs) const
{
  validateMod(rhs);
  return this->number > rhs.number;
}
bool Modulo::operator>=(const Modulo &rhs) const
{
  validateMod(rhs);
  return this->number >= rhs.number;
}
bool Modulo::operator<(const Modulo &rhs) const
{
  validateMod(rhs);
  return this->number < rhs.number;
}
bool Modulo::operator<=(const Modulo &rhs) const
{
  validateMod(rhs);
  return this->number <= rhs.number;
}
bool Modulo::operator==(const Modulo &rhs) const
{
  return this->mod == rhs.mod && this->number == rhs.number;
}

std::ostream &operator<<(std::ostream &out, const Modulo &modulo)
{
  out << modulo.number << ' ' << modulo.mod;
  return out;
}
std::istream &operator>>(std::istream &in, Modulo &modulo)
{
  in >> modulo.number >> modulo.mod;
  return in;
}
```

main.cpp – содержит точку входа в программу, а так же логику для интерактивного взаимодействия с классом Modulo.

```
#include <iostream>
#include <limits>
#include <string>
#include "modulo.hpp"

/*
 Сравнивает два экземпляра класса Modulo.
 Результат сравнения выводит в стандартный поток вывода.
*/
void cmpModulo(const Modulo &lhs, const Modulo &rhs)
{
  try
  {
    if (lhs > rhs)
      std::cout << "lhs is bigger then rhs" << std::endl;
    else if (lhs == rhs)
      std::cout << "lhs is equal rhs" << std::endl;
    else if (lhs < rhs)
      std::cout << "lhs is less than rhs" << std::endl;
  }
  catch (std::string err)
  {
    std::cout << err << std::endl;
  }
}

bool readVariables(Modulo &lhs, Modulo &rhs)
{
  if (std::cin >> lhs >> rhs)
    return true;

  std::cin.clear();
  std::cin.ignore(std::numeric_limits<std::streamsize>::max(), '\n');
  std::cout << "Переменные введены неправильно" << std::endl;
  return false;
```

```
}

int main()
{
  // Обработка ввода команд
  std::string str; // Считанная команда
  while (std::cin >> str)
  {
    if (str == "exit")
      exit(0);
    else if (str == "help")
    {
      std::cout << "Опернды вводить в следующем формате:\n"
                   "\t<число> <модуль>\n"
                   "Доступные команды:\n"
                   "\texit    - выйти из приложения"
                   "\thelp    - вывести справку об командах"
                   "\tadd a b - сложить a и b\n"
                   "\tsub a b - вычесть b из a\n"
                   "\tmul a b - умножить a на b\n"
                   "\tsub a b - разделить a на b\n"
                   "\tcmp a b - сравнить a и b\n";
    }

    try
    {
      Modulo lhs;
      Modulo rhs;
      if (str == "add")
      {
        if (readVariables(lhs, rhs))
          std::cout << lhs + rhs << '\n';
      }
      else if (str == "sub")
      {
        if (readVariables(lhs, rhs))
          std::cout << lhs - rhs << '\n';
```

```
      }
      else if (str == "mul")
      {
        if (readVariables(lhs, rhs))
          std::cout << lhs * rhs << '\n';
      }
      else if (str == "div")
      {
        if (readVariables(lhs, rhs))
        {
          if (rhs.GetNumber() == 0)
            std::cout << "Нельзя делить на ноль" << std::endl;
          else
            std::cout << lhs / rhs << '\n';
        }
      }
      else if (str == "cmp")
      {
        if (readVariables(lhs, rhs))
          cmpModulo(lhs, rhs);
      }
      else
      {
        std::cin.ignore(std::numeric_limits<std::streamsize>::max(),
'\n');
        std::cout << "Такой команды не существует\n";
      }
    }
    catch (char const *err)
    {
      std::cout << err << std::endl;
    }
  }
}
```

GitHub

Я хочу хранить все лабораторные работы в одном репозитории. Ссылка на лабораторную работу: https://github.com/Reterer/course_oop/tree/master/Lab1

Тестирование

Все тесты можно хранить в одном файле, так как программа работает в интерактивном режиме. Однако для удобства я разделил тесты на наборы по смыслу. Было составлено 3 набора тестов:

- Тестирование операций и ввода команд (файл test_01.txt)

```
Тестовые данные        ->    Правильный ответ
add 5 101 10 101       ->    15 101
sub 10 101 5 101       ->    5 101
mul 5 101 10 101       ->    50 101
div 10 101 5 101       ->    2 101

add 100 101 2 101      ->    1 101
sub 120 101 18 101     ->    1 101
mul 10  101 11 101     ->    9 101

cmp 10 101 5 101       ->    lhs is bigger then rhs
cmp 5  101 5 101       ->    lhs is equal rhs
cmp 5 101 10 101       ->    lhs is less than rhs
```

- Тестирование ввода неправильной команды (ввод команд не идеален, но считаю, его достаточно, чтобы работать с классом Modulo)

```
Тестовые данные        ->    Правильный ответ
fsdjfsdhfs             ->    Такой команды не существует
```

- Тестирование некорректных входных данных

```
Тестовые данные        ->    Правильный ответ
add 5 101 6 100        ->    diff modulo
add 5 0   6 0          ->    zero modulo
div 5 101 0 101        ->    Нельзя делить на ноль
```

Для более-менее автоматического тестирования я написал скрипт test.sh на bash, который последовательно запускает программу на каждом тестовом наборе а затем сравнивает результат работы с правильным ответом. Скрипт находит все тестовые наборы в папке tests. Но при этом выполняющийся файл программы должен находится в папке ./build относительно test.sh и называться oop_exercise_01.

Объяснение результатов работы программы

Задание: Создать класс Modulo для работы с целыми числами по модулю N. В классе должно быть два поля: число и N. Реализовать все арифметические операции. Реализовать операции сравнения.
Честно говоря, постановка задания очень расплывчатая. Так как работать с числами по модулю можно разными способами (кто сколько придумает).

Поэтому я определил арифметические операции следующем образом:

- Сложение – как сложение по модулю N
- Вычитание — как вычитание по модулю N
- Умножение – как умножение по модулю N
- Деление – как обычное деление, а затем взятие по модулю N

Операции сравнения определены, как стандартные по полю number (число).
При этом эти операции работают только с операндами, у которых одинаковый модуль. В противном случае возникает ошибка «diff modulo». Ошибка «zero modulo» возникает, если пытаться задать в конструкторе поле modulo равным нулю.

Программа работает в интерактивном режиме. Доступны следующие команды:

- help – выводит справку по командам
- exit – завершает работу приложения
- add <Modulo> <Modulo> – складывает два экземпляра класса Modulo;
- sub <Modulo> <Modulo> – вычитает из 1-го поеранда 2-й;
- mul <Modulo> <Modulo> – умножает два операнда;
- div <Modulo> <Modulo> – делит 1-й операнд на 2-й;
- cmp <Modulo> <Modulo> – сравнивает два аргумента между собой;

<Modulo> вводится следующим образом: <number> <module>, то есть вводится поле number и поле module соответственно.

Давайте разберем работу программы на примере выполнения команды cmp.

1. Программа, после запуска, входит в цикл while с условием «пока поток ввода открыт»
2. Программа ожидает ввода слова (первой части команды) из потока cin в переменную str. После ввода слова, программа переходит к следующему шагу
3. Выполняется сопоставление введенного слова с условиями блоков if/else if, которые и выполняют заданную команду. Если не было найдено нужного условия, выполняется блок else (main.cpp:79), который выводит сообщение «Такой команды не существует». Если введенная команда была exit, то происходит выход из программы путем вызова функции exit.

На 49 строке происходит инициализация переменных rhs и lhs, которые хранят значения введенных аргументов. Для ввода/вывода значений экземпляра класса Modulo были операторы << и >> для работы с стандартными потоками. Для выполнения арифметических операций и операций сравнения так же были перегружены соответствующие операторы.

Выводы

В данной лабораторной работе я использовал классы и перегрузку стандартных операторов. Использование классов помогает строить абстракции и скрывать определение этих объектов. А перегрузка операторов позволяет приблизить пользовательские типы к встроенным.

Выполняя эту лабораторную работу я узнал о перегрузке операторов. А так же применил новые для меня функции стандартного потока ввода.

Список литературы

1. Язык программирования C++. Краткий курс, 2-е изд. : Пер. с англ. - СПб.: ООО "Диалектика", 2019. - 320 с.: ил. - Парал. тит. англ.
2. Обработка некорректного ввода через std::cin [Электронный ресурс]. URL:https://ravesli.com/urok-72-obrabotka-nekorrektnogo-vvoda-cherez-std-cin/ (дата обращения: 18.09.2020).