**федеральное государственное бюджетное образовательное учреждение высшего образования «Московский авиационный институт (национальный исследовательский университет)»**Институт “Информационные технологии и прикладная математика”

**Лабораторная работа № 1**

по курсу «ООП»

Тема: Простые классы на языке С++

Студент: Панкратова Ксения

Группа: 80-201

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2021

1. **Постановка задачи**

Вариант 13

Создать класс Long для работы с целыми без знаковыми числами из 64 бит. Число должно быть представлено двумя полями unsigned int. Должны быть реализованы арифметические операции, присутствующие в C++ , и сравнения.

1. **Решение**

Репозиторий на GitHub: <https://github.com/sechensy/op_exercise_01>

Для решения задачи необходимо создать класс Long, в котором будет храниться 2 unsigned int

Long(const unsigned int first, const unsigned int second)  
{  
 this-> first = first;  
 this-> second = second;  
}

Конструктором класс является функция Long.

Для вывода и ввода объектов данного класса необходимо переопределить операторы.

//переопределение оператора вывода  
 friend std::ostream& operator<<(std::ostream& os, const Long& p){  
 return os<<p.getFirst()<<"|"<<p.getSecond();  
 }  
//переопределение оператора ввода  
 friend std::istream& operator>>(std::istream& is, Long& p){  
 is>>p.first>>p.second;  
 return is;  
 }

Методы getFirst и getSecond возвращают поля класса Long

//метод возвращающий первое поле  
 unsigned int getFirst()const  
 {  
 return first;  
 }  
//метод возвращающий второе поле  
 unsigned int getSecond()const  
 {  
 return second;  
 }

Методы setFirst и setSecond устанавливают значения в поля

//метод устанавливающий первое поле  
 void setFirst(unsigned int first)  
 {  
 this->first=first;  
 }  
//метод устанавливающий второе поле  
 void setSecond(unsigned int second)  
 {  
 this->second=second;  
 }

Чтобы выполнить поставленную задачу, связанную с реализацией арифметических операций нам нужно перегрузить эти функции(вычитания, сложения, умножения, деления, остатка от деления, а так же сравнения)

Перегрузка Сложения(первое число одного объекта нужно будет сложить с первым числом второго, также и для вторых чисел )

//перегрузка сложения  
 friend Long operator+(const Long& d1, const Long& d2)  
 {  
 return Long(d1.getFirst()+d2.getFirst(), d1.getSecond() + d2.getSecond());  
 }

Перегрузка вычитания

//перегрузка вычитания  
 friend Long operator-(const Long &d1, const Long &d2)  
 {  
 // max число unsigned int 4 294 967 295;  
 //4294967295 Никак не получить при нормальном вычитание  
 if(d1.getSecond()<d2.getSecond()) {//тут проверка вторых чисел (с первыми легче и поэтому они проверяются сразу в main)  
 return Long(4294967295, 4294967295);  
 }else {  
 //для тех моментов когда вычитание делается правильно  
 return Long(d1.getFirst() - d2.getFirst(), d1.getSecond() - d2.getSecond());  
 }  
 }

В if() проверяю только вторые поля у каждого из объектов, потому что проверка первых чисел реализована в перегрузке операторов сравнения (они будут дальше), а дальше результатом устанавливается максимальное число(его мы никак не можем получить при обычном вычитании), а уже в самом main() идет проверка: что сумма чисел которые мы вычитаем должна быть больше их разности, и если это не так, то в консоли пишется ошибка , Но если при вычитании второго объекта из первого, у первого первое число и второе больше чем у второго объекта, то в дело вступает else{}

Дальше идет перегрузка операторов умножения, деления и ост от деления

//перегрузка умножения  
 friend Long operator\*(const Long &d1, const Long &d2)  
 {  
 return Long(d1.getFirst() \* d2.getFirst(), d1.getSecond() \* d2.getSecond());  
 }  
//перегрузка деления  
 friend Long operator/(const Long &d1, const Long &d2)  
 {  
 return Long(d1.getFirst() / d2.getFirst(), d1.getSecond() / d2.getSecond());  
 }  
//перегрузка ост от дел  
 friend Long operator%(const Long &d1, const Long &d2)  
 {  
 return Long(d1.getFirst() % d2.getFirst(), d1.getSecond() % d2.getSecond());  
 }

Тут почти ничем не отличается от суммы(тут вместо + -> \*,/,%), так что не расписываю

Операторы сравнения:

//перегрузка меньше  
 friend bool operator<(const Long &d1, const Long &d2)  
 {  
 return (d1.getFirst() < d2.getFirst() && d1.getSecond() < d2.getSecond()) ||  
 (d1.getFirst() < d2.getFirst() && d1.getSecond() > d2.getSecond()) ||  
 (d1.getFirst() < d2.getFirst() && d1.getSecond() == d2.getSecond()) ||  
 (d1.getFirst() == d2.getFirst() && d1.getSecond() < d2.getSecond());  
 }  
//перегрузка больше  
 friend bool operator>(const Long &d1, const Long &d2)  
 {  
 return (d1.getFirst() > d2.getFirst() && d1.getSecond() < d2.getSecond()) ||  
 (d1.getFirst() > d2.getFirst() && d1.getSecond() > d2.getSecond()) ||  
 (d1.getFirst() > d2.getFirst() && d1.getSecond() == d2.getSecond()) ||  
 (d1.getFirst() == d2.getFirst() && d1.getSecond() > d2.getSecond());  
 }  
//перегрузка равенства  
 friend bool operator==(const Long &d1, const Long &d2)  
 {  
 return d1.getFirst() == d2.getFirst(), d1.getSecond() == d2.getSecond();  
 }

Реализовано почти так же, как и в обычной математике, но у объекта этого класса может быть такое число, например Long(1,1000)

Общий смысл сравнения: если первое число одного объекта больше чем у другого объекта, то оно и больше; если первые числа у них одинаковые, то сравниваются они уже по второй цифре, также как с первыми числами(если у 1-ого объект больше второе число, то он и больше), эта логика работает и с меньше, а при равенстве два числа первого объекта должны быть равны двум числам второго.

Main():

Есть бесконечный цикл, из него можно выйти при написании “stop”, этот цикл запрашивает числа для создания двух объектов класса, а затем проводит с ними вычисления, а также сравнивает их

1. **Тесты**
2. Введите значения для первого числа:   
   2  
   0  
   Введите значения для второго числа:   
   1  
   0  
   me0ow1= 2|0 me0ow2= 1|0  
   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
   сложение(Long) = (3|0)  
   вычитание(Long) = (1|0)  
   умножение(Long) = (2|0)  
   деление(Long) = (2|0)  
   остаток от деления(Long) = (0|0)  
   сравнение(Long) = (2|0>1|0)  
   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
   если хотите провести еще один тест, то напишите любое слови или букву, если нет, то stop   
   next  
   Введите значения для первого числа:   
   2  
   0  
   Введите значения для второго числа:   
   1  
   1  
   me0ow1= 2|0 me0ow2= 1|1  
   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
   сложение(Long) = (3|1)  
   ошибка при вычитании  
   умножение(Long) = (2|0)  
   деление(Long) = (2|0)  
   остаток от деления(Long) = (0|0)  
   сравнение(Long) = (2|0>1|1)  
   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
   если хотите провести еще один тест, то напишите любое слови или букву, если нет, то stop   
   next  
   Введите значения для первого числа:   
   5  
   0  
   Введите значения для второго числа:   
   7  
   0  
   me0ow1= 5|0 me0ow2= 7|0  
   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
   сложение(Long) = (12|0)  
   ошибка при вычитании  
   умножение(Long) = (35|0)  
   деление(Long) = (0|0)  
   остаток от деления(Long) = (5|0)  
   сравнение(Long) = (5|0<7|0)  
   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
   если хотите провести еще один тест, то напишите любое слови или букву, если нет, то stop   
   next  
   Введите значения для первого числа:   
   8  
   1  
   Введите значения для второго числа:   
   9  
   5  
   me0ow1= 8|1 me0ow2= 9|5  
   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
   сложение(Long) = (17|6)  
   ошибка при вычитании  
   умножение(Long) = (72|5)  
   деление(Long) = (0|0)  
   остаток от деления(Long) = (8|1)  
   сравнение(Long) = (8|1<9|5)  
   \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
   если хотите провести еще один тест, то напишите любое слови или букву, если нет, то stop   
   stop  
     
   Process finished with exit code 0
3. **Листинг программы**

\*Панкратова Ксения М80-201Б-20  
Вариант 13  
Создать класс Long для работы с целыми без знаковыми числами из 64 бит.  
Число должно быть представлено двумя полями unsigned int.  
Должны быть реализованы арифметические операции, присутствующие в C++ , и сравнения.  
\*/

#include <iostream>  
#include <string>  
  
  
using namespace std;  
class Long  
{  
protected:  
//поля класса Long  
 unsigned int first;  
 unsigned int second;  
public:  
// конструктор с двумя значениями на вход  
 Long(const unsigned int first, const unsigned int second)  
 {  
 this-> first = first;  
 this-> second = second;  
 }  
//деконструктор  
 ~Long()=default;  
//метод возвращающий первое поле  
 unsigned int getFirst()const  
 {  
 return first;  
 }  
//метод возвращающий второе поле  
 unsigned int getSecond()const  
 {  
 return second;  
 }  
//метод устанавливающий первое поле  
 void setFirst(unsigned int first)  
 {  
 this->first=first;  
 }  
//метод устанавливающий второе поле11  
 void setSecond(unsigned int second)  
 {  
 this->second=second;  
 }  
////<это переопределение при котором при вычитании 1-2 получается 1>  
// friend Long operator-(const Long &d1, const Long &d2)  
// {  
// if(d1.getFirst() < d2.getFirst() && d1.getSecond() < d2.getSecond()){  
// return Long(d2.getFirst() - d1.getFirst(), d2.getSecond() - d1.getSecond());  
//  
// } else if(d1.getFirst() < d2.getFirst() && d1.getSecond() > d2.getSecond()){  
// return Long(d2.getFirst() - d1.getFirst(), d1.getSecond() - d2.getSecond());  
//  
// } else if(d1.getFirst() > d2.getFirst() && d1.getSecond() < d2.getSecond()){  
// return Long(d1.getFirst() - d2.getFirst(), d2.getSecond() - d1.getSecond());  
//  
// }else if(d1.getFirst() > d2.getFirst() && d1.getSecond() > d2.getSecond()){  
// return Long(d1.getFirst() - d2.getFirst(), d1.getSecond() - d2.getSecond());  
// }else{  
// return Long(d1.getFirst() - d2.getFirst(), d1.getSecond() - d2.getSecond());  
// }  
// }  
////</это переопределение при котором при вычитании 1-2 получается 1>  
  
//перегрузка вычитания  
 friend Long operator-(const Long &d1, const Long &d2)  
 {  
 // max число unsigned int 4 294 967 295;  
 //4294967295 Никак не получить при нормальном вычитание  
 if(d1.getSecond()<d2.getSecond()) {//тут проверка вторых чисел (с первыми легче и поэтому они проверяются сразу в main)  
 return Long(4294967295, 4294967295);  
 }else {  
 //для тех моментов когда вычитание делается правильно  
 return Long(d1.getFirst() - d2.getFirst(), d1.getSecond() - d2.getSecond());  
 }  
 }  
  
//перегрузка сложения  
 friend Long operator+(const Long& d1, const Long& d2)  
 {  
 return Long(d1.getFirst()+d2.getFirst(), d1.getSecond() + d2.getSecond());  
 }  
//перегрузка умножения  
 friend Long operator\*(const Long &d1, const Long &d2)  
 {  
 return Long(d1.getFirst() \* d2.getFirst(), d1.getSecond() \* d2.getSecond());  
 }  
//перегрузка деления  
 friend Long operator/(const Long &d1, const Long &d2)  
 {  
 return Long(d1.getFirst() / d2.getFirst(), d1.getSecond() / d2.getSecond());  
 }  
//перегрузка ост от дел  
 friend Long operator%(const Long &d1, const Long &d2)  
 {  
 return Long(d1.getFirst() % d2.getFirst(), d1.getSecond() % d2.getSecond());  
 }  
  
//перегрузка меньше  
 friend bool operator<(const Long &d1, const Long &d2)  
 {  
 return (d1.getFirst() < d2.getFirst() && d1.getSecond() < d2.getSecond()) ||  
 (d1.getFirst() < d2.getFirst() && d1.getSecond() > d2.getSecond()) ||  
 (d1.getFirst() < d2.getFirst() && d1.getSecond() == d2.getSecond()) ||  
 (d1.getFirst() == d2.getFirst() && d1.getSecond() < d2.getSecond());  
 }  
//перегрузка больше  
 friend bool operator>(const Long &d1, const Long &d2)  
 {  
 return (d1.getFirst() > d2.getFirst() && d1.getSecond() < d2.getSecond()) ||  
 (d1.getFirst() > d2.getFirst() && d1.getSecond() > d2.getSecond()) ||  
 (d1.getFirst() > d2.getFirst() && d1.getSecond() == d2.getSecond()) ||  
 (d1.getFirst() == d2.getFirst() && d1.getSecond() > d2.getSecond());  
 }  
//перегрузка равенства  
 friend bool operator==(const Long &d1, const Long &d2)  
 {  
 return d1.getFirst() == d2.getFirst(), d1.getSecond() == d2.getSecond();  
 }  
//переопределение оператора вывода(оператор сдвига <<)  
 friend std::ostream& operator<<(std::ostream& os, const Long& p){  
 return os<<p.getFirst()<<"|"<<p.getSecond();  
 }  
//переопределение оператора ввода(оператор сдвига >>)  
 friend std::istream& operator>>(std::istream& is, Long& p){  
 is>>p.first>>p.second;  
 return is;  
 }  
};  
int main(){  
 string a = "nonStop";  
 while(true){  
 cout<<"Введите значения для первого числа: "<<endl;  
 Long me0ow1(0,0);//дефолтные значения(ниже расписано)  
 cin>> me0ow1;  
 cout<<"Введите значения для второго числа: "<<endl;  
 Long me0ow2(0,0);//дефолтные значения  
 cin>> me0ow2;  
 cout<<"me0ow1= "<< me0ow1<<" me0ow2= "<<me0ow2<<endl;//вывод данных с которыми будудут производится рассчеты  
 cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl;  
 cout<<"сложение(Long) = ("<<me0ow1+me0ow2<<")"<<endl;  
 if(me0ow1+me0ow2<me0ow1-me0ow2){//если вы найдете числа у которых сумма меньше их разности (именно в unsigned) то тут все сломается  
 cout<<"ошибка при вычитании"<<endl;  
 } else {  
 cout<<"вычитание(Long) = ("<<me0ow1-me0ow2<<")"<<endl;  
 }  
 cout<<"умножение(Long) = ("<<me0ow1\*me0ow2<<")"<<endl;  
 cout<<"деление(Long) = ("<<me0ow1/me0ow2<<")"<<endl;  
 cout<<"остаток от деления(Long) = ("<<me0ow1%me0ow2<<")"<<endl;  
 if (me0ow1>me0ow2){  
 cout<<"сравнение(Long) = ("<<me0ow1 << ">" << me0ow2 << ")"<< endl;  
 }  
 else if (me0ow1<me0ow2){  
 cout<<"сравнение(Long) = ("<<me0ow1 << "<" << me0ow2 << ")"<< endl;  
 }  
 else {  
 cout<<"сравнение(Long) = ("<<me0ow1 << "==" << me0ow2 << ")"<< endl;  
 }  
 cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl;  
 cout<<"если хотите провести еще один тест, то напишите любое слови или букву, если нет, то stop "<<endl;  
 cin>>a;  
 if(a=="Stop" || a=="stop" || a=="STOP") break;  
 else continue;  
 }  
  
////мои тесты(проверка работоспособности ввода и вывода)  
////Long me0ow1(0,0);//дефолтные значения, которые не повлияют ни на что,  
//// но это мой костыль(даже если реально будет объект с 0|0 то ничего не поменяется)  
// cin>>me0ow1;//просто тест работы вывода и ввода  
// cout<<me0ow1;

}

1. **Литература**

1. Справочник по языку С++ [Электронный ресурс].

URL: https://docs.microsoft.com/ru-ru/cpp/cpp/cpp-language-reference?view=msvc-160 (дата обращения: 15.09.2021).