Московский Авиационный Институт

(Национальный Исследовательский Университет)

Кафедра 806 «Вычислительная информатика и программирование»

Факультет: «Информационные технологии и прикладная математика»

Лабораторная работа

Дисциплина: «Объектно-ориентированное программирование»

III семестр

Задание 2: «Операторы, литералы»

|  |  |
| --- | --- |
| Группа: | М8О-206Б-18, №13 |
| Студент: | Макаренкова Вера Михайловна |
| Преподаватель: | Журавлёв Андрей Андреевич |
| Оценка: |  |
| Дата: | 22.11.2019 |

Москва, 2019

**Задание**

13.Создать класс Long для работы с целыми беззнаковыми числами из 64 бит. Число должно быть

представлено двумя полями unsigned int. Должны быть реализованы арифметические операции,

присутствующие в С++, и сравнения с помощью перегрузки операторов.

**Адрес репозитория на GitHub**

https://github.com/vera0000/oop\_exercise\_02

**Код программы на С++**

CMakeLists.txt

cmake\_minimum\_required(VERSION 3.2)

project(Long)

add\_executable(Long

Source.cpp

Long.cpp)

set\_property(TARGET Long PROPERTY CXX\_STANDART 11)

Long.cpp

#include "Long.h"

#include <stdlib.h>

#include <iostream>

#include <string>

#include <vector>

#include <climits>

#include <exception>

#include <stdexcept>

unsigned long long grade(unsigned long long m, int n){

int i;

unsigned long long rez = 1;

for(int i = 1 ; i <= n; i++) {

rez \*= m;

}

return rez;

}

Long::Long() {

firstHalf = 0;

secondHalf = 0;

}

Long::Long(const char \* in) : Long() {

std::string str = std::string(in);

std::string sec(str.size(), '0');

std::vector<int> v;

while (str != sec) {

int a = 0;

for (int i = 0; i < str.size(); i++) {

a \*= 10;

a += str[i] - '0';

str[i] = char('0' + a / 2);

a %= 2;

}

v.push\_back(a);

}

unsigned long long sh = 1;

for (int i = 0; i < 32 && i < v.size(); i++) {

secondHalf += v[i] \* sh;

sh \*= 2;

}

unsigned long long fh = 1;

for (int i = 32; i < v.size(); i++) {

firstHalf += v[i] \* fh;

fh \*= 2;

}

}

Long::Long(std::string str) : Long() {

std::string sec(str.size(), '0');

std::vector<int> v;

while (str != sec) {

int a = 0;

for (int i = 0; i < str.size(); i++) {

a \*= 10;

a += str[i] - '0';

str[i] = char('0' + a / 2);

a %= 2;

}

v.push\_back(a);

}

unsigned long long sh = 1;

for (int i = 0; i < 32 && i < v.size(); i++) {

secondHalf += v[i] \* sh;

sh \*= 2;

}

unsigned long long fh = 1;

for (int i = 32; i < v.size(); i++) {

firstHalf += v[i] \* fh;

fh \*= 2;

}

}

Long Long::operator+(const Long &l2) const{

Long l;

unsigned int add = 0;

Long zero;

if ((UINT\_MAX - secondHalf) < l2.secondHalf) {

l.secondHalf = l2.secondHalf - (UINT\_MAX - secondHalf);

add += 1;

} else {

l.secondHalf = secondHalf + l2.secondHalf;

}

if ((UINT\_MAX - firstHalf) < l2.firstHalf) {

std::cout << "Error. Int overflow\n";

return zero;

} else {

l.firstHalf = firstHalf + l2.firstHalf;

if (l.firstHalf == UINT\_MAX && add > 0){

std::cout << "Error. Int overflow\n";

return zero;

} else {

l.firstHalf += add;

}

}

return l;

}

Long Long::operator-(const Long &l2) const{

Long l;

unsigned long long union1 = 0;

unsigned long long union2 = 0;

long long union0 = 0;

union1 = secondHalf + (firstHalf \* grade(2, 32));

union2 = l2.secondHalf + (l2.firstHalf \* grade(2, 32));

if (union1 > union2) {

union0 = union1 - union2;

}

else if(union1 < union2){

union0 = union2 - union1;

}

l.secondHalf = union0 % (grade(2, 32));

l.firstHalf = union0 / (grade(2, 32));

return l;

}

Long Long::operator\*(const Long &l2) const{

Long l;

unsigned long long union1 = 0;

unsigned long long union2 = 0;

unsigned long long union0 = -1;

Long zero;

union1 = secondHalf + (firstHalf \* grade(2, 32));

union2 = l2.secondHalf + (l2.firstHalf \* grade(2, 32));

if ((l2 == zero) || (firstHalf == 0 && secondHalf ==0)){

return zero;

}

union0 = union1 \* union2;

l.secondHalf = union0 % (grade(2, 32));

l.firstHalf = union0 / (grade(2, 32));

}

return l;

}

Long Long::operator/(const Long &l2) const{

Long l;

unsigned long long union1 = 0;

unsigned long long union2 = 0;

long long union0 = -1;

Long zero;

union1 = secondHalf + (firstHalf \* grade(2, 32));

union2 = l2.secondHalf + (l2.firstHalf \* grade(2, 32));

if(union2 != 0){

union0 = union1 / union2;

}

else if(union2 == 0){

std::cout << "Error. Cannot be divided by zero\n";

return zero;

}

if (union0 != -1){

l.secondHalf = union0 % (grade(2, 32));

l.firstHalf = union0 / (grade(2, 32));

}

return l;

}

Long Long::operator%(const Long &l2) const{

Long l;

unsigned long long union1 = 0;

unsigned long long union2 = 0;

long long union0 = -1;

Long zero;

union1 = secondHalf + (firstHalf \* grade(2, 32));

union2 = l2.secondHalf + (l2.firstHalf \* grade(2, 32));

if(union2 != 0){

union0 = union1 % union2;

}

else if(union2 == 0){

std::cout << "Error. Cannot be divided by zero\n";

return zero;

}

if (union0 != -1){

l.secondHalf = union0 % (grade(2, 32));

l.firstHalf = union0 / (grade(2, 32));

}

return l;

}

void Long::operator++(){

Long l;

Long one;

one.secondHalf++;

unsigned int add = 0;

Long zero;

if ((UINT\_MAX - secondHalf) < one.secondHalf) {

l.secondHalf = one.secondHalf - (UINT\_MAX - secondHalf);

add += 1;

} else {

l.secondHalf = secondHalf + one.secondHalf;

}

if ((UINT\_MAX - firstHalf) < one.firstHalf) {

std::cout << "Error. Int overflow\n";

firstHalf = zero.firstHalf;

secondHalf = zero.secondHalf;

return;

} else {

l.firstHalf = firstHalf + one.firstHalf;

if (l.firstHalf == UINT\_MAX && add > 0) {

std::cout << "Error. Int overflow\n";

firstHalf = zero.firstHalf;

secondHalf = zero.secondHalf;

return;

} else {

l.firstHalf += add;

}

}

firstHalf = l.firstHalf;

secondHalf = l.secondHalf;

}

void Long::operator--(){

Long l, l1;

Long zero;

Long one;

one.secondHalf++;

l1.firstHalf = firstHalf;

l1.secondHalf = secondHalf;

if(one > l1){

std::cout << "Error. -- Cannot be calculated \n";

return;

}

if ((one < l1) || (one == l1)){

l1 = l1 - one;

}

firstHalf = l1.firstHalf;

secondHalf = l1.secondHalf;

}

bool Long::operator==(Long &l2) const{

return ((firstHalf==l2.secondHalf) && (secondHalf==l2.secondHalf));

}

bool Long::operator>(Long &l2) const{

return ((firstHalf > l2.firstHalf) || (firstHalf == l2.firstHalf && secondHalf > l2.secondHalf));

}

bool Long::operator<(Long &l2) const{

return ((firstHalf < l2.firstHalf) || (firstHalf == l2.firstHalf && secondHalf < l2.secondHalf));

}

Long operator""\_long(const char\* str){

return Long(str);

}

void Long::print(std::ostream &os) const{

Long l1;

l1.firstHalf = firstHalf;

l1.secondHalf = secondHalf;

std::vector<int> v;

while (l1.firstHalf != 0) {

v.push\_back(l1.firstHalf % 2);

l1.firstHalf /= 2;

}

for (int i = 0; i < 32 - v.size(); i++) {

std::cout << 0;

}

for (int i = v.size() - 1; i >= 0; i--) {

std::cout << v[i];

}

v.clear();

std::cout << " ";

while (l1.secondHalf != 0) {

v.push\_back(l1.secondHalf % 2);

l1.secondHalf /= 2;

}

for (int i = 0; i < 32 - v.size(); i++) {

std::cout << 0;

}

for (int i = v.size() - 1; i >= 0; i--) {

std::cout << v[i];

}

std::cout << '\n';

}

std::istream& operator>> (std::istream& is, Long& l2) {

std::string a;

is >> a;

l2 = Long(a);

}

std::ostream& operator<< (std::ostream& os, const Long& l2) {

l2.print(os);

}

Long.h

#ifndef \_\_Long\_h\_\_

#define \_\_Long\_h\_\_

#include <iostream>

#include <string>

class Long

{

public:

Long();

Long(const char \*);

Long(std::string);

void read(std::istream &is);

Long operator+(const Long &l2) const;

Long operator-(const Long &l2) const;

Long operator\*(const Long &l2) const;

Long operator/(const Long &l2) const;

Long operator%(const Long &l2) const;

void operator++();

void operator--();

bool operator==(Long &l2) const;

bool operator>(Long &l2) const;

bool operator<(Long &l2) const;

void print(std::ostream &os) const;

unsigned int firstHalf;

unsigned int secondHalf;

};

Long operator""\_long(const char\* str);

std::istream& operator>>(std::istream& is, Long& l2);

std::ostream& operator<<(std::ostream& os, const Long& l2);

unsigned long long grade(unsigned long long m, int n);

#endif

Source.cpp

#include "Long.h"

int main(int argc, char\*\* argv){

Long l1, l2;

std::cout << "Enter the number\n";

std::cin >> l1;

std::cout << "Enter the second number\n";

std::cin >> l2;

std::cout << l1;

std::cout <<l2;

std::cout << "Sum is :\n";

Long lFinal = l1 + l2;

std::cout << lFinal;

std::cout << "Differ is:\n";

lFinal = l1 - l2;

std::cout << lFinal;

std::cout << "Multiplication is:\n";

lFinal = l1 \* l2;

std::cout << lFinal;

std::cout << "Division is:\n";

lFinal = l1 / l2;

std::cout << lFinal;

std::cout << "Remainder from division is:\n";

lFinal = l1 % l2;

std::cout << lFinal;

lFinal = l1;

std::cout << "++ of the first is:\n";

++lFinal;

std::cout << lFinal;

lFinal = l2;

std::cout << "-- of the second is:\n";

--lFinal;

std::cout << lFinal;

if (l1 > l2) {

std::cout << "First number is larger\n";

} else if (l1 < l2) {

std::cout << "Second number is larger\n";

} else {

std::cout << "First and second numbers are equal\n";

}

std::cout << "Literal examples 89\_long and 0\_long: \n";

std::cout << 89\_long;

std::cout << 0\_long;

Long l;

std::cout << "Enter the number for demonstration of input and output\n";

std::cin >> l;

std::cout << l;

return 0;

}

Результаты тестов

1

Enter the number

4294967296

Enter the second number

4567

00000000000000000000000000000001 00000000000000000000000000000000

00000000000000000000000000000000 00000000000000000001000111010111

Sum is :

00000000000000000000000000000001 00000000000000000001000111010111

Differ is:

00000000000000000000000000000000 11111111111111111110111000101001

Multiplication is:

Error.Overflow

00000000000000000000000000000000 00000000000000000000000000000000

Division is:

00000000000000000000000000000000 00000000000011100101100110010011

Remainder from division is:

00000000000000000000000000000000 00000000000000000000001010001011

++ of the first is:

00000000000000000000000000000001 00000000000000000000000000000001

-- of the second is:

00000000000000000000000000000000 00000000000000000001000111010110

First number is larger

Literal examples 89\_long and 0\_long:

00000000000000000000000000000000 00000000000000000000000001011001

00000000000000000000000000000000 00000000000000000000000000000000

Enter the number for demonstration of input and output

7

00000000000000000000000000000000 00000000000000000000000000000111

2

Enter the number

72718199239305465356465465456554564

Enter the second number

5

00010010111000000011001110000110 00110011011111110111001001000100

00000000000000000000000000000000 00000000000000000000000000000101

Sum is :

00010010111000000011001110000110 00110011011111110111001001001001

Differ is:

00010010111000000011001110000110 00110011011111110111001000111111

Multiplication is:

Error.Overflow

00000000000000000000000000000000 00000000000000000000000000000000

Division is:

00000011110001100111000010110100 01110000101100110001011011011010

Remainder from division is:

00000000000000000000000000000000 00000000000000000000000000000010

++ of the first is:

00010010111000000011001110000110 00110011011111110111001001000101

-- of the second is:

00000000000000000000000000000000 00000000000000000000000000000100

First number is larger

Literal examples 89\_long and 0\_long:

00000000000000000000000000000000 00000000000000000000000001011001

00000000000000000000000000000000 00000000000000000000000000000000

Enter the number for demonstration of input and output

6

00000000000000000000000000000000 00000000000000000000000000000110

3

Enter the number

0

Enter the second number

67

00000000000000000000000000000000 00000000000000000000000000000000

00000000000000000000000000000000 00000000000000000000000001000011

Sum is :

00000000000000000000000000000000 00000000000000000000000001000011

Differ is:

00000000000000000000000000000000 00000000000000000000000001000011

Multiplication is:

00000000000000000000000000000000 00000000000000000000000000000000

Division is:

00000000000000000000000000000000 00000000000000000000000000000000

Remainder from division is:

00000000000000000000000000000000 00000000000000000000000000000000

++ of the first is:

00000000000000000000000000000000 00000000000000000000000000000001

-- of the second is:

00000000000000000000000000000000 00000000000000000000000001000010

Second number is larger

Literal examples 89\_long and 0\_long:

00000000000000000000000000000000 00000000000000000000000001011001

00000000000000000000000000000000 00000000000000000000000000000000

Enter the number for demonstration of input and output

890

00000000000000000000000000000000 00000000000000000000001101111010

Объяснение результатов

Программа получает на вход два числа, далее они преобразуются в 64-битовое представление и выполняет требуемые задание лабораторной работы.

Вывод

В данной лабораторной работе были изучены операторы и литералы, которые при работе могут значительно уменьшить количество кода, а так же сделать его более понятным и лаконичным.