

Московский Авиационный Институт
(Национальный Исследовательский Университет)

Кафедра 806 «Вычислительная информатика и программирование»
Факультет: «Информационные технологии и прикладная математика»

Лабораторная работа
Дисциплина: «Объектно-ориентированное программирование»
II семестр
Задание 1: «Операторы, литералы»

Москва, 2019

1.Тема: Операторы, литералы

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

1.Задание (вариант 7):

Создать класс для работы с 128-битовыми строками. Битовая строка должна быть представлена двумя полями типа `uint64_t`. Должны быть реализованы все традиционные операции для работы с битами: `and`, `or`, `xor`, `not`. Реализовать сдвиг влево и сдвиг вправо на заданное количество битов. Реализовать операцию вычисления количества единичных битов, операцию сравнения по количеству единичных битов и операцию проверки включения. Операции должны быть выполнены в виде перегрузки операторов.

1.Адрес репозитория на GitHub https://github.com/DragonKeker/oop_exercise_02

1.Код программы на C++

logika.h

```
#pragma once
#ifndef D_LOGIKA_H_
#define D_LOGIKA_H_ 1
#include <iostream>

struct logika {

    logika operator&(const logika &obj2)const;
    logika operator|(const logika &obj2)const;
    logika operator^(const logika &obj2)const;
    logika operator~()const;
    logika operator<<(const int k);
    logika operator>>(const int k);
    int _count1()const;
    void _sravnenie( logika& obj2);
    bool operator>(const logika obj2)const;
    bool operator<(const logika obj2)const;
    bool operator==(const logika& obj2)const;
    bool _vkluch(const logika& obj2)const;

public:
    uint64_t field1;
    uint64_t field2;
    static void read(std::istream&, logika&);
    void write(std::ostream&);
};

    void operator ""_out(const char* str, size_t size);
    logika operator "" _l(const char* str);

#endif // D_LOGIKA_H_#pragma once
```

logika.cpp

```
#include "logika.h"
#include <bitset>
#include <assert.h>
/* void logika::read(std::istream& is, logika& obj) {
    is >> obj.field1 >> obj.field2;
}
void logika::write(std::ostream& os) {
```

```

        os << field1 << " " << field2 << "\n";
    }*/

logika logika::operator&(const logika& obj2)const
{
    logika result;
    result.field1 = field1 & obj2.field1;
    result.field2 = field2 & obj2.field2;
    return(result);
};
logika logika::operator|(const logika& obj2)const
{
    logika result;
    result.field1 = this->field1 | obj2.field1;
    result.field2 = this->field2 | obj2.field2;
    return(result);
};
logika logika::operator^(const logika& obj2)const
{
    logika result;
    result.field1 = this->field1 ^ obj2.field1;
    result.field2 = this->field2 ^ obj2.field2;
    return(result);
};
logika logika::operator~()const
{
    logika result;
    result.field1 = ~field1;
    result.field2 = ~field2;
    return(result);
};
logika logika::operator<<(int k)
{
    logika result;
    uint64_t a = field1, b = field2;
    unsigned long long pow63 = 1;
    for (int i = 0; i < 63; i++) {
        pow63 *= 2;
    }

    for (int i = 0; i < k; i++) {
        a = a << 1;
        if (b >= pow63) {
            a += 1;
        }
        b = b << 1;
    }
    result.field1 = a;
    result.field2 = b;
    return (result);
};
logika logika::operator>>(int k)
{
    logika result;
    uint64_t a = field1, b = field2;

    unsigned long long pow63 = 1;
    for (int i = 0; i < 63; i++) {
        pow63 *= 2;
    }
    for (int i = 0; i < k; i++) {
        b = b >> 1;
        if (a % 2 == 1) {
            b += pow63;
        }
    }
    result.field1 = a;
    result.field2 = b;
    return (result);
};

```

```

        }
        a = a >> 1;
    }

    result.field1 = a;
    result.field2 = b;
    return (result);
};

int logika::_count1()const
{
    uint64_t a = field1;
    uint64_t b = field2;
    int t = 0;
    int sum = 0;
    while (a != 0) {
        t += 1; a &= a - 1;
    }
    sum = sum + t;

    t = 0;
    while (b != 0) {
        t += 1; b &= b - 1;
    }
    sum = sum + t;

    return(sum);
};

bool logika::operator>(const logika obj2)const
{
    if (this->_count1() > obj2._count1())
    {
        return(1);
    }
    else
        return(0);
};

bool logika::operator<(const logika obj2)const
{
    if (this->_count1() < obj2._count1())
    {
        return(1);
    }
    else
        return(0);
};

bool logika::operator==(const logika& obj2)const
{
    if (this->_count1() == obj2._count1())
    {
        return(1);
    }
    else
        return(0);
};

void logika::_sravnenie( logika& obj2)
{

```

```

        if (*this>(obj2))
        {
            std::cout<< "Количество единиц в первой строке больше" << std::endl;
        }
        else if (*this<(obj2))
        {
            std::cout << "Количество единиц во второй строке больше"<< std::endl;
        }
        else
        {
            std::cout<< "Количество единиц в строках равно" << std::endl;
        }
    };

    bool logika::_vkluch(const logika& obj2)const {

        if ((obj2.field1 == this->operator&(obj2).field1) && (obj2.field2 == this-
>operator&(obj2).field2))
            return (1);
        else
            return 0;
    };

    void operator ""_out(const char* str, size_t size) {
        std::cout << str << std::endl;
    }

    logika operator ""_l(const char* str)
    {
        int cnt=0, j=0;
        logika result1;
        unsigned long long result = 0;
        char str1[65]="0", str2[65] = "0";

        size_t size = strlen(str);
        for (j = 0; j < strlen(str); j++) {
            if ((strlen(str) - j) > 64) {
                str1[j] = str[j];
            }
            else { str2[cnt] = str[j];
                cnt++;
            }
        };

        size_t size1 = strlen(str1), size2 = strlen(str2);

        for (size_t i = 0; i < size1; ++i)
        {
            assert(str1[i] == '1' || str1[i] == '0');
            result |= (str1[i] - '0') << (size1 - i - 1);
        }

        result1.field1 = result;
        result = 0;

        for (size_t k = 0; k < size2; ++k)
        {
            assert(str2[k] == '1' || str2[k] == '0');
            result |= (str2[k] - '0') << (size2 - k - 1);
        }

        result1.field2 = result;

        return result1;
    }

```


Ввод k = 1

Ввод k = 1

Ожидаемый результат

1 0

7 1

6 1

18446744073709551608 18446744073709551615

Введите : k 1

14 0

Введите : k 1

3 9223372036854775808

Всего единиц в первой строке: 3

Количество единиц в первом числе больше

Не включает

2 0

2 тест

test_02

Ввод

1 1

1 1

Ввод k = 2

Ввод k = 2 1 1

1 1

0 0

18446744073709551614 18446744073709551614

Введите : k 2

4 4

Введите : k 2

0 4611686018427387904

Всего единиц в первой строке: 2

Количество единиц в числах равно

Включает

2 0

3 тест test_03 Ввод

1 7

5 5

Ввод k = 3

Ввод k = 11 5

5 7

4 2

18446744073709551614 18446744073709551608

Введите : k 3

8 56

Введите : k 1

0 9223372036854775811

Всего единиц в первой строке: 4

Количество единиц в числах равно

Не включает

2 0

1.Результаты выполнения тестов

Введите данные

7 0

1 1

1 0

7 1

6 1

18446744073709551608 18446744073709551615

Введите : k 1

14 0

Введите : k 1

3 9223372036854775808

Всего едениц в первой строке: 3

Количество единиц в первом числе больше

Не включает

2 0

C:\Users\Андрей\source\repos\Совершенная 1 лаба\Debug\Совершенная 1 лаба.exe (процесс 17360) завершает работу с кодом 0.

Чтобы автоматически закрывать консоль при остановке отладки, установите параметр "Сервис" -> "Параметры" -> "Отладка" -> "Автоматически закрыть консоль при остановке отладки".

Чтобы закрыть это окно, нажмите любую клавишу...

Введите данные

1 1

1 1

1 1

1 1

0 0

18446744073709551614 18446744073709551614

Введите : k 2

4 4

Введите : k 2

0 4611686018427387904

Всего едениц в первой строке: 2

Количество единиц в числах равно

Включает

2 0

C:\Users\Андрей\source\repos\Совершенная 1 лаба\Debug\Совершенная 1 лаба.exe (процесс 18128) завершает работу с кодом 0.

Чтобы автоматически закрывать консоль при остановке отладки, установите параметр "Сервис" -> "Параметры" -> "Отладка" -> "Автоматически закрыть консоль при остановке отладки".

Чтобы закрыть это окно, нажмите любую клавишу...

Введите данные

1 7

5 5

1 5

5 7

4 2

18446744073709551614 18446744073709551608

Введите : k 3

8 56

Введите : k 1

0 9223372036854775811

Всего едениц в первой строке: 4

Количество единиц в числах равно

Не включает

2 0

C:\Users\Андрей\source\repos\Совершенная 1 лаба\Debug\Совершенная 1 лаба.exe (процесс 16044) завершает работу с кодом 0.

Чтобы автоматически закрывать консоль при остановке отладки, установите параметр "Сервис" -> "Параметры" -> "Отладка" -> "Автоматически закрыть консоль при остановке отладки".

Чтобы закрыть это окно, нажмите любую клавишу...

1.Объяснение результатов работы программы - вывод

Применение перегрузки операторов может существенно облегчить и ускорить процесс написания кода, однако, может и запутать код.

Пользовательские литералы позволяют создавать объекты пользовательского типа посредством суффикса. Они позволяют работать с строками, символами и численными значениями. Их использование может как повысить читаемость кода и упростить его написание.