**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: Егорова Анна (староста)

Группа: 80-207

Преподаватель: Чернышов Л.Н.

Дата:

Оценка:

Москва, 2019

1. Постановка задачи:

Создать класс BitString для работы с 96-битовыми строками. Битовая строка должна быть представлена

двумя полями: старшая часть unsigned long long, младшая часть unsigned int. Должны быть

реализованы все традиционные операции для работы с битами: and, or, xor, not. Реализовать сдвиг влево

shiftLeft и сдвиг вправо shiftRight на заданное количество битов. Реализовать операцию вычисления

количества единичных битов, операции сравнения по количеству единичных битов. Реализовать операцию

проверки включения.

1. Описание программы:

Программа состоит из описания класса, который состоит из двух переменных типа unsigned long long и unsigned int, а также двух переменных этого же типа для хранения результата операций. Кроме это класс содержит функции логических операций. В функции main вводятся значения полей и выводятся результаты всех операций, описанных в задании.

Битовые операции в языке:

1) & - and

2) | - or

3) << - shiftleft

4) >> - shiftright

5) ^ - xor

6) ~ - not

Далее программа выполняет подсчёт единиц в выбранной строке, сравнение строк по количеству единиц и проверку на включение.

Для подсчёта количества единиц делаем цикл, который увеличивает счётчик и умножает число на меньшее его на единицу(тем самым обнуляется младшая единица), пока само не станет равным нулю.

Для сравнения по количеству единиц сравниваем результаты выполнения предыдущий функции для обоих чисел. Для того, чтобы проверить, является ли число Б включением А, нужно проверить не стоит ли у Б единицы в тех местах, где у А - 0. Делаем логическое сложение и сравниваем с А.

После выполнения операций, программа выводит в стандартный вывод либо результирующие строки в бинарном виде(строка 96 бит, поля разделяются пробелом), либо результат выполнения функций.

3. Набор testcases:

|  |  |  |
| --- | --- | --- |
|  | Входный данные\*: | Выходные данные\*\*: |
| 1 тест: | 4  3  2  1 | 1. 0 1 2. 6 3 3. 6 2 4. 5 4 3 2 5. 2 1 1 0 6. 8 6 4 2 7. 1 8. 1 9. 2 10. 1 11. Первое число больше второго по количеству единичных битов. 12. Не включено. 13. Включено. 14. Выход из программы |
| 2 тест: | 23456  345  12345  234 | 1. 4128 72 2. 31673 507 3. 27545 435 4. 23456 346 12346 235 5. 11728 172 6172 117 6. 46912 690 24690 468 7. 7 8. 6 9. 5 10. 5 11. Первое число больше второго по количеству единичных битов. 12. Не включено. 13. Не включено. 14. Выход из программы |
| 3 тест: | 98765  54  12763  4567 | 1. 457 22 2. 111071 4599 3. 110614 4577 4. 98766 55 12764 4568 5. 49382 27 6381 2283 6. 197530 108 25526 9134 7. 8 8. 9 9. 4 10. 8 11. Первое число меньше второго по количеству единичных битов. 12. Не включено. 13. Не включено. 14. Выход из программы |

\* 1 число: старшая часть 1 строки

2 число: младшая часть 1 строки

3 число: старшая часть 2 строки

4 число: младшая часть 2 строки

\*\* Все ответы должны быть представлены в 10 системе счисления и 2 системе счисления.

4. Результаты выполнения тестов.

**Тест 1:**

Старшая часть 1 строки:

4

Младшая часть 1 строки:

3

Старшая часть 2 строки:

2

Младшая часть 2 строки:

1

Нажмите (1) для операции and с битами

Нажмите (2) для операции or с битами

Нажмите (3) для операции xor с битами

Нажмите (4) для операции not с битами

Нажмите (5) для сдвига вправо

Нажмите (6) для сдвига влево

Нажмите (7) для подсчета единичных битов в старшей части 1 числа

Нажмите (8) для подсчета единичных битов в старшей части 2 числа

Нажмите (9) для подсчета единичных битов в младшей части 1 числа

Нажмите (10) для подсчета единичных битов в младшей части 2 числа

Нажмите (11) для равнения по количеству единичных битов

Нажмите (12) для проверки на включение старшей части

Нажмите (13) для проверки на включение младшей части

Нажмите (14) если хотите выйти

1

0 1

00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000

00000000 00000000 00000000 00000001

2

6 3

00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000110

00000000 00000000 00000000 00000011

3

6 2

00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000110

00000000 00000000 00000000 00000010

4

5 4

00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000101

00000000 00000000 00000000 00000100

3 2

00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000011

00000000 00000000 00000000 00000010

5

4 >> 1 == 2

00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000010

3 >> 1 == 1

00000000 00000000 00000000 00000001

2 >> 1 == 1

00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000001

1 >> 1 == 0

00000000 00000000 00000000 00000000

6

4 << 1 == 8

00000000 00000000 00000000 00000000 00000000 00000000 00000000 00001000

3 << 1 == 6

00000000 00000000 00000000 00000110

2 << 1 == 4

00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000100

1 << 1 == 2

00000000 00000000 00000000 00000010

7

00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000100

1

8

00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000010

1

9

00000000 00000000 00000000 00000011

2

10

00000000 00000000 00000000 00000001

1

11

Первое число больше второго по количеству единичных битов.

12

Не включено.

13

Включено.

14

**Тест 2:**

1

4128 72

00000000 00000000 00000000 00000000 00000000 00000000 00010000 00100000

00000000 00000000 00000000 01001000

2

31673 507

00000000 00000000 00000000 00000000 00000000 00000000 01111011 10111001

00000000 00000000 00000001 11111011

3

27545 435

00000000 00000000 00000000 00000000 00000000 00000000 01101011 10011001

00000000 00000000 00000001 10110011

4

23457 346

00000000 00000000 00000000 00000000 00000000 00000000 01011011 10100001

00000000 00000000 00000001 01011010

12346 235

00000000 00000000 00000000 00000000 00000000 00000000 00110000 00111010

00000000 00000000 00000000 11101011

5

23456 >> 1 == 11728

00000000 00000000 00000000 00000000 00000000 00000000 00101101 11010000

345 >> 1 == 172

00000000 00000000 00000000 10101100

12345 >> 1 == 6172

00000000 00000000 00000000 00000000 00000000 00000000 00011000 00011100

234 >> 1 == 117

00000000 00000000 00000000 01110101

6

23456 << 1 == 46912

00000000 00000000 00000000 00000000 00000000 00000000 10110111 01000000

345 << 1 == 690

00000000 00000000 00000010 10110010

12345 << 1 == 24690

00000000 00000000 00000000 00000000 00000000 00000000 01100000 01110010

234 << 1 == 468

00000000 00000000 00000001 11010100

7

00000000 00000000 00000000 00000000 00000000 00000000 01011011 10100000

7

8

00000000 00000000 00000000 00000000 00000000 00000000 00110000 00111001

6

9

00000000 00000000 00000001 01011001

5

10

00000000 00000000 00000000 11101010

5

11

Первое число больше второго по количеству единичных битов.

12

Не включено.

13

Не включено.

14

**3 тест:**

Старшая часть 1 строки:

98765

Младшая часть 1 строки:

54

Старшая часть 2 строки:

12763

Младшая часть 2 строки:

4567

Нажмите (1) для операции and с битами

Нажмите (2) для операции or с битами

Нажмите (3) для операции xor с битами

Нажмите (4) для операции not с битами

Нажмите (5) для сдвига вправо

Нажмите (6) для сдвига влево

Нажмите (7) для подсчета единичных битов в старшей части 1 числа

Нажмите (8) для подсчета единичных битов в старшей части 2 числа

Нажмите (9) для подсчета единичных битов в младшей части 1 числа

Нажмите (10) для подсчета единичных битов в младшей части 2 числа

Нажмите (11) для равнения по количеству единичных битов

Нажмите (12) для проверки на включение старшей части

Нажмите (13) для проверки на включение младшей части

Нажмите (14) если хотите выйти

1

457 22

00000000 00000000 00000000 00000000 00000000 00000000 00000001 11001001

00000000 00000000 00000000 00010110

2

111071 4599

00000000 00000000 00000000 00000000 00000000 00000001 10110001 11011111

00000000 00000000 00010001 11110111

3

110614 4577

00000000 00000000 00000000 00000000 00000000 00000001 10110000 00010110

00000000 00000000 00010001 11100001

4

98766 55

00000000 00000000 00000000 00000000 00000000 00000001 10000001 11001110

00000000 00000000 00000000 00110111

12764 4568

00000000 00000000 00000000 00000000 00000000 00000000 00110001 11011100

00000000 00000000 00010001 11011000

5

98765 >> 1 == 49382

00000000 00000000 00000000 00000000 00000000 00000000 11000000 11100110

54 >> 1 == 27

00000000 00000000 00000000 00011011

12763 >> 1 == 6381

00000000 00000000 00000000 00000000 00000000 00000000 00011000 11101101

4567 >> 1 == 2283

00000000 00000000 00001000 11101011

6

98765 << 1 == 197530

00000000 00000000 00000000 00000000 00000000 00000011 00000011 10011010

54 << 1 == 108

00000000 00000000 00000000 01101100

12763 << 1 == 25526

00000000 00000000 00000000 00000000 00000000 00000000 01100011 10110110

4567 << 1 == 9134

00000000 00000000 00100011 10101110

7

00000000 00000000 00000000 00000000 00000000 00000001 10000001 11001101

8

8

00000000 00000000 00000000 00000000 00000000 00000000 00110001 11011011

9

9

00000000 00000000 00000000 00110110

4

10

00000000 00000000 00010001 11010111

8

11

Первое число меньше второго по количеству единичных битов.

12

Не включено.

13

Не включено.

5. Листинг программы:

anyaegorovaa@LAPTOP-9PIFT2VJ:~/2kurs/oop/lab1$ cat **CMakeLists.txt**

project(lab1)

add\_executable(oop\_exercise\_01 main.cpp)

set(CMAKE\_CXX\_FLAGS -Wall)

anyaegorovaa@LAPTOP-9PIFT2VJ:~/2kurs/oop/lab1$ cat **main.cpp**

//Егорова Анна М8О-107Б-18

//Создать класс BitString для работы с 96-битовыми строками. Битовая строка должна быть представлена

//двумя полями: старшая часть unsigned long long, младшая часть unsigned int. Должны быть

//реализованы все традиционные операции для работы с битами: and, or, xor, not. Реализовать сдвиг влево

//shiftLeft и сдвиг вправо shiftRight на заданное количество битов. Реализовать операцию вычисления

//количества единичных битов, операции сравнения по количеству единичных битов. Реализовать операцию

//проверки включения.

#include <iostream>

#include <cmath>

using namespace std;

class BitString{

public: unsigned long long m1;

unsigned int m2;

unsigned long long n1;

unsigned int n2;

unsigned long long o;

};

int perevod(unsigned long long m1) {

const int size = sizeof(unsigned long long)\*8;

unsigned long long i;

int binNum[size] = {0};

for (i = 0; i < size; i++) {

binNum[i] = m1 % 2;

m1 /= 2;

}

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

cout << binNum[i];

if (i%8 == 0)

cout << " ";

}

cout << "\n";

}

int perevod2(unsigned int n1) {

const int size = sizeof(unsigned int)\*8;

unsigned int i;

int binNum[size] = {0};

for (i = 0; i < size; i++) {

binNum[i] = n1 % 2;

n1 /= 2;

}

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

cout << binNum[i];

if (i%8 == 0)

cout << " ";

}

cout << "\n";

}

int main() {

BitString a;

cout << "Старшая часть 1 строки:" << endl;

cin >> a.m1;

cout << "Младшая часть 1 строки:" << endl;

cin >> a.n1;

BitString b;

cout << "Старшая часть 2 строки:" << endl;

cin >> a.m2;

cout << "Младшая часть 2 строки:" << endl;

cin >> a.n2;

int kol, koll, kol2, koll2;

int key = 0;

while(key != 14) {

cout << "Нажмите (1) для операции and с битами" <<endl

<< "Нажмите (2) для операции or с битами" << endl

<< "Нажмите (3) для операции xor с битами " << endl

<< "Нажмите (4) для операции not с битами" <<endl

<< "Нажмите (5) для сдвига вправо" << endl

<< "Нажмите (6) для сдвига влево" <<endl

<< "Нажмите (7) для подсчета единичных битов в старшей части 1 числа" <<endl

<< "Нажмите (8) для подсчета единичных битов в старшей части 2 числа" <<endl

<< "Нажмите (9) для подсчета единичных битов в младшей части 1 числа" <<endl

<< "Нажмите (10) для подсчета единичных битов в младшей части 2 числа" <<endl

<< "Нажмите (11) для равнения по количеству единичных битов" <<endl

<< "Нажмите (12) для проверки на включение старшей части" <<endl

<< "Нажмите (13) для проверки на включение младшей части" <<endl

<< "Нажмите (14) если хотите выйти"<<endl;

cin >> key;

switch(key) {

case 1: {

unsigned long long c1 = a.m1 & a.m2;

unsigned int c2 = a.n1 & a.n2; //and

printf("%llu %u\n", c1, c2);

perevod(c1);

perevod2(c2);

break;}

case 2: {

unsigned long long d1 = a.m1 | a.m2;

unsigned int d2 = a.n1 | a.n2; //or

printf("%llu %u\n", d1, d2);

perevod(d1);

perevod2(d2);

break;}

case 3: {

unsigned long long e1 = a.m1 ^ a.m2;

unsigned int e2 = a.n1 ^ a.n2; //xor

printf("%llu %u\n", e1, e2);

perevod(e1);

perevod2(e2);

break;}

case 4: {

int u1 = ~a.m1;

int u2 = ~a.n1;

int u3 = ~a.m2;

int u4 = ~a.n2; //not

printf("%d %d\n", -u1, -u2);

perevod(-u1);

perevod2(-u2);

printf("%d %d\n", -u3, -u4);

perevod(-u3);

perevod2(-u4);

break;}

case 5: {

unsigned long long y1 = a.m1, p1;

unsigned int y2 = a.n1, p2;

unsigned long long y3 = a.m2, p3;

unsigned int y4 = a.n2, p4;

printf("%llu >> 1 == %llu\n", y1, y1 >> 1);

p1 = (y1 >> 1);

perevod(p1);

printf("%u >> 1 == %u\n", y2, y2 >> 1);

p2 = (y2 >> 1);

perevod2(p2);

printf("%llu >> 1 == %llu\n", y3, y3 >> 1);

p3 = (y3 >> 1);

perevod(p3);

printf("%u >> 1 == %u\n", y4, y4 >> 1);

p4 = (y4 >> 1);

perevod2(p4);

break;}

case 6: {

unsigned long long y1 = a.m1, q1;

unsigned int y2 = a.n1, q2;

unsigned long long y3 = a.m2, q3;

unsigned int y4 = a.n2, q4;

printf("%llu << 1 == %llu\n", y1, y1 << 1);

q1 = (y1 << 1);

perevod(q1);

printf("%u << 1 == %u\n", y2, y2 << 1);

q2 = (y2 << 1);

perevod2(q2);

printf("%llu << 1 == %llu\n", y3, y3 << 1);

q3 = (y3 << 1);

perevod(q3);

printf("%u << 1 == %u\n", y4, y4 << 1);

q4 = (y4 << 1);

perevod2(q4);

break;}

case 7: {

const int size = sizeof(unsigned long long)\*8;

unsigned long long i;

kol = 0;

int binNum[size] = {0};

for (i = 0; i < size; i++) {

binNum[i] = a.m1 % 2;

a.m1 /= 2;

}

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

{

cout << binNum[i];

if (binNum[i] == 1) {

kol++;}

if (i%8 == 0)

cout << " ";

}

cout << "\n" << kol << endl;

break; }

case 8: {

const int size = sizeof(unsigned long long)\*8;

unsigned long long i;

koll = 0;

int binNum[size] = {0};

for (i = 0; i < size; i++) {

binNum[i] = a.m2 % 2;

a.m2 /= 2;

}

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

cout << binNum[i];

if (binNum[i] == 1) {

koll++;}

if (i%8 == 0)

cout << " ";

}

cout << "\n" << koll << endl;

break; }

case 9: {

const int size = sizeof(unsigned int)\*8;

unsigned int j;

kol2 = 0;

int binNum[size] = {0};

for (j = 0; j < size; j++)

{

binNum[j] = a.n1 % 2;

a.n1 /= 2; }

for (int j = size - 1; j >= 0; j--)

{

cout << binNum[j];

if (binNum[j] == 1) {

kol2++;}

if (j%8 == 0)

cout << " ";

}

cout << "\n" << kol2 << endl;

break; }

case 10: {

const int size = sizeof(unsigned int)\*8;

unsigned int j;

koll2 = 0;

int binNum[size] = {0};

for (j = 0; j < size; j++)

{

binNum[j] = a.n2 % 2;

a.n2 /= 2; }

for (int j = size - 1; j >= 0; j--)

{

cout << binNum[j];

if (binNum[j] == 1) {

koll2++;}

if (j%8 == 0)

cout << " ";

}

cout << "\n" << koll2 << endl;

break; }

case 11: {

if ((kol+kol2) > (koll+koll2)) {

cout << "Первое число больше второго по количеству единичных битов." << endl;

}

if ((kol+kol2) < (koll+koll2)) {

cout << "Первое число меньше второго по количеству единичных битов."<< endl;

}

if ((kol+kol2) == (koll+koll2)) {

cout << "Первое число равно второму по количеству единичных битов." << endl;

}

break; }

case 12: { if ((a.m1 | a.m2) == a.m1) {

cout << "Включено." << endl; }

else if ((a.m1 | a.m2) != a.m1) {

cout << "Не включено." << endl; }

break; }

case 13: {

if ((a.n1 | a.n2) == a.n1) {

cout << "Включено." << endl; }

else if ((a.n1 | a.n2) != a.n1) {

cout << "Не включено." << endl; }

break; }

}

}

return 0;

}

6. Вывод:

Во время лабораторной работы я познакомилась с языком С++ и создала простейший класс на нём. В отличие от языка Си функции пишутся внутри класса, что намного удобнее. Кроме того в С++ есть модификаторы доступа: public — доступ открыт всем, кто видит определение данного класса, private — доступ открыт самому классу (т.е. функциям-членам данного класса) и друзьям (friend) данного класса - как функциям, так и классам. Однако производные классы не получают доступа к этим данным совсем. И все другие классы такого доступа не имеют.

7. Список литературы.

1. Побитовые операции С++ [Электронный ресурс]. URL: https://ravesli.com/urok-45-pobitovye-operatory/