



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных
технологий

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 1

по дисциплине

«Структуры и алгоритмы обработки данных»

Тема: «Поразрядные операции и их применение»

Выполнил студент группы ИКБО-15-22

Оганнисян Г.А.

Принял преподаватель

Лабораторная работа выполнена

«_»_____2023 г.

(подпись студента)

«Зачтено»

«_»_____2023 г.

(подпись руководителя)

Москва 2022

Цель.

Получить навыки применения поразрядных операций в алгоритмах.

Личный вариант

Вариант №22

Номер бита	Номер бита	Множитель	Делитель	Задание для выражения
Только с нечетными номерами	5-ой, 3-ый, 11-ый	8	16	Обнулить n -ый бит, используя маску (вар 1)

1. Задание 1

1.1. Первый пункт задания

1.1.1. Условие:

Определить переменную целого типа, присвоить ей значение, используя константу в шестнадцатеричной системе счисления. Разработать оператор присваивания и его выражение, которое установит заданные в задании биты (Только с нечетными номерами) исходного значения переменной в значение 1, используя соответствующую маску и поразрядную операцию.

1.1.2. Выражение реализующие операцию:

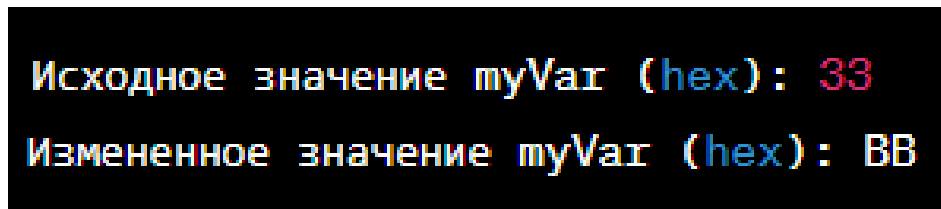
`myVar |= mask;`

Маска определяется при инициализации переменной: `int mask = 0xAA;`

1.1.3. Код функции, реализующей задание первого пункта, 22 варианта:

```
int main() {
    int myVar;
    myVar = 0x33;
    cout << "Iskhodnoe znachenie myVar (hex): " << hex << myVar << endl;
    int mask = 0xAA;
    myVar |= mask;
    cout << "Resultat myVar (hex): " << hex << myVar << endl;
    return 0;
}
```

1.1.4. Результаты тестирования:



```
Исходное значение myVar (hex): 33
Измененное значение myVar (hex): BВ
```

Рисунок 1. – Результаты тестирования кода 1.1.3

1.2. Второй пункт задания

1.2.1. Условие:

Определить переменную целого типа. Разработать оператор присваивания и его выражение, которое обнуляет заданные в задании биты (5-ой, 3-ый, 11-ый) исходного значения переменная, используя соответствующую маску и поразрядную операцию. Значение в переменную вводится с клавиатуры.

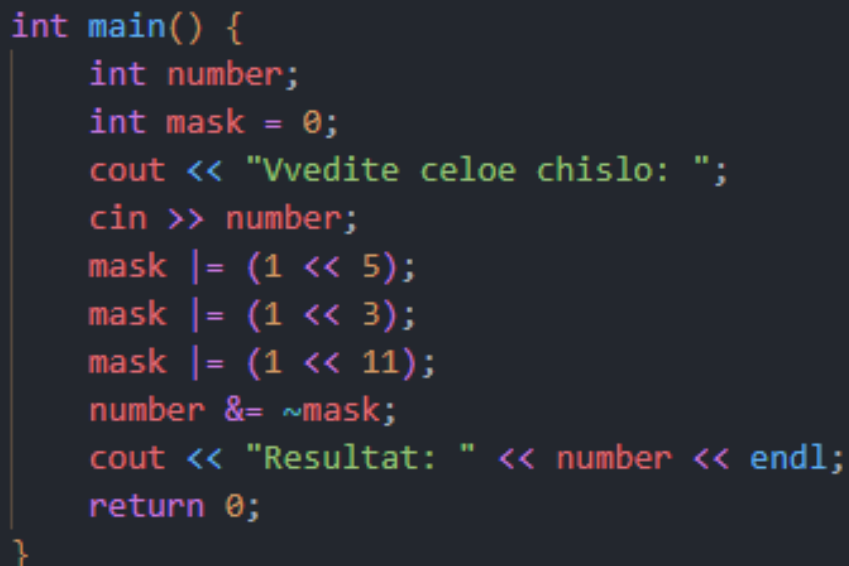
1.2.2. Выражение реализующие операцию:

`number &= ~mask;`

формирование маски: `mask |= (1 << 5); mask |= (1 << 3);`

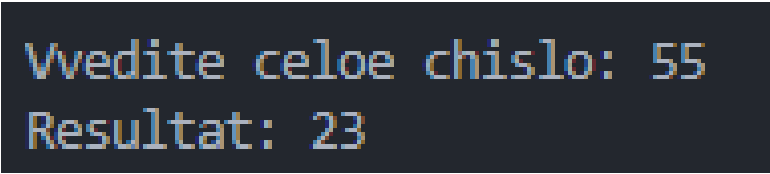
`mask |= (1 << 11);`

1.2.3. Код функции, реализующей задание второго пункта, 22 варианта:



```
int main() {
    int number;
    int mask = 0;
    cout << "Vvedite celoe chislo: ";
    cin >> number;
    mask |= (1 << 5);
    mask |= (1 << 3);
    mask |= (1 << 11);
    number &= ~mask;
    cout << "Resultat: " << number << endl;
    return 0;
}
```

1.2.4. Результаты тестирования:



```
Vvedite celoe chislo: 55
Resultat: 23
```

Рисунок 2. – Результаты тестирования кода 1.2.3

1.3. Третий пункт задания

1.3.1. Условие:

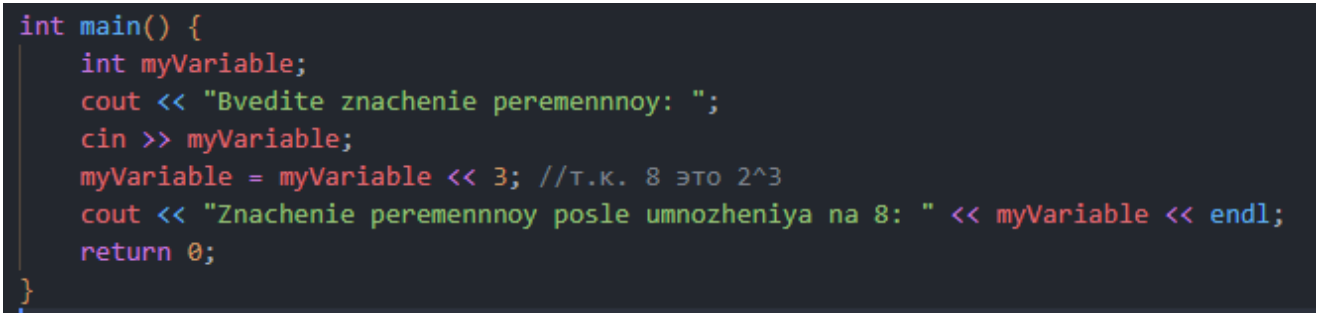
1.3.2.

Определить переменную целого типа. Разработать оператор присваивания и выражение, которое умножает значение переменной на число, указанное в третьем столбце варианта (8), используя соответствующую поразрядную операцию. Изменяемое число вводится с клавиатуры.

1.3.3. Выражение реализующие операцию:

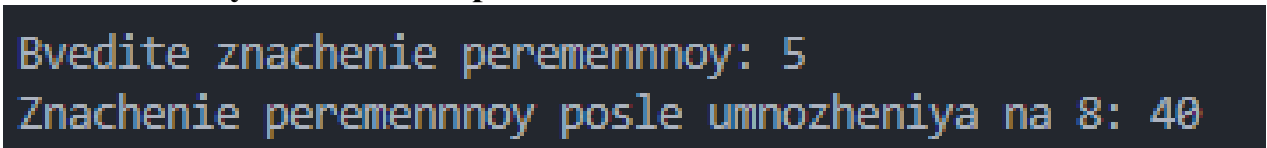
`myVariable = myVariable << 3`

1.3.4. Код функции, реализующей задание третьего пункта, 22 варианта:



```
int main() {
    int myVariable;
    cout << "Bvedite znachenie peremennnoy: ";
    cin >> myVariable;
    myVariable = myVariable << 3; //т.к. 8 это 2^3
    cout << "Znachenie peremennnoy posle umnozheniya na 8: " << myVariable << endl;
    return 0;
}
```

1.3.5. Результаты тестирования:



```
Bvedite znachenie peremennnoy: 5
Znachenie peremennnoy posle umnozheniya na 8: 40
```

Рисунок 3. – Результаты тестирования кода 1.3.4

1.4. Четвертый пункт задания

1.4.1. Условие:

Определить переменную целого типа. Разработать оператор присваивания и выражение, которое делит значение переменной на число, указанное в четвертом столбце варианта (16), используя соответствующую поразрядную операцию. Изменяемое число вводится с клавиатуры.

1.4.2. Выражение реализующие операцию:

`myVariable = myVariable >> 4;`

1.4.3. Код функции, реализующей задание четвертого пункта, 22 варианта

```
int main() {  
    int myVariable;  
    cout << "Vvedite znachenie peremennnoy: ";  
    cin >> myVariable;  
    int delitel = 8;  
    myVariable = myVariable >> 4; // //т.к. 16 это 2^4  
    cout << "Znachenie peremennnoy posle deleniya na 16: " << myVariable << endl;  
    return 0;  
}
```

1.4.4. Результаты тестирования:

```
Vvedite znachenie peremennnoy: 54  
Znachenie peremennnoy posle deleniya na 16: 3
```

Рисунок 4. – Результаты тестирования кода 1.4.3

1.5. Пятый пункт задания

1.5.1. Условие:

Определить переменную целого типа. Разработать оператор присваивания и выражение, в котором используются только поразрядные операции. В выражении используется маска – переменная. Маска может быть инициализирована единицей в младшем разряде (вар 1) или единицей в старшем разряде (вар 2). Изменяемое число вводится с клавиатуры (*Обнулить n -ый бит, используя маску (вар 1)*).

1.5.2. Выражение реализующие операцию:

`mask = ~mask;`

маска формируется в выражении из пункта соответствующей константы: `int mask = 1 << n;`

1.5.3. Код функции, реализующей задание пятого пункта, 22 варианта:

```
int main() {
    int myVariable;
    cout << "Vvedite znachenie peremennoy: ";
    cin >> myVariable;
    int n;
    cout << "Vvedite nomer bite, kotoryy nujno obnulit: ";
    cin >> n;
    int mask = 1 << n;
    mask = ~mask;
    myVariable = myVariable & mask;
    cout << "Znachenie peremennoy posle obnuleniya " << n << "-go bita: " << myVariable << endl;
    return 0;
}
```

1.5.4. Результаты тестирования:

```
Vvedite znachenie peremennoy: 56201451
Vvedite nomer bite, kotoryy nujno obnulit: 6
Znachenie peremennoy posle obnuleniya 6-go bita: 56201387
```

Рисунок 5. – Результаты тестирования кода 1.5.3

2. Задание 2

2.1. Постановка задачи:

Реализовать задачу по сортировке данных файла, используя для представления данных файла (10^7 семизначных чисел) в памяти, массив битов.

2.2. Алгоритм решения:

- 1) Создать булев массив (битовый массив) bitArray размером 10 000 000 (для чисел от 0 до 9999999).
- 2) Считывать семизначные числа с клавиатуры (пока не получено достаточное количество чисел).
- 3) Для каждого считанного числа:
- 4) Установить соответствующий бит в bitArray в значение true.
- 5) Пройти по bitArray и вывести отсортированные числа

2.3. Тестовый пример, демонстрирующий входные данные и заполненный битовый массив (не более 20 чисел).

```
Vvedite kolichество chisel: 20
Generaciya:
6158962 5436844 4114610 5413790 6736556 3585613 5748 8125161 2034717 8470714 740312 1081524 880198 7211645 1345348 8478745 6684867 6495611
6925319 7356116
Otsortirovannye chisla:
5748 740312 880198 1081524 1345348 2034717 3585613 4114610 5413790 5436844 6158962 6495611 6684867 6736556 6925319 7211645 7356116 8125161
8470714 8478745
Vremya raboti: 850342 mc
```

Рисунок 6. – Результаты тестирования функции sortirovka

2.4. Код программы

```
int main() {
    const int maxNum = 9999999;
    int numCount;
    cout << "Vvedite kolichество chisel: ";
    cin >> numCount;
    if (numCount <= 0) {
        cout << "Oshibka.\n";
        return 1;
    }
    vector<bool> bitArray(maxNum + 1, false);
    vector<int> randomNumbers;
    random_device rd;
    mt19937 gen(rd());
    uniform_int_distribution<int> dis(0, maxNum);
    auto start = chrono::high_resolution_clock::now();
    for (int i = 0; i < numCount; ++i) {
        int num = dis(gen);
        randomNumbers.push_back(num);
        bitArray[num] = true;
    }
    cout << "Generaciya:\n";
    for (int num : randomNumbers) {
        cout << num << " ";
    }
    cout << "\n";
    cout << "Otsortirovannye chisla:\n";
    for (int i = 0; i <= maxNum; ++i) {
        if (bitArray[i]) {
            cout << i << " ";
        }
    }
    cout << "\n";
    auto stop = chrono::high_resolution_clock::now();
    auto duration = chrono::duration_cast<chrono::microseconds>(stop - start);
    cout << "Vremya raboti: " << duration.count() << " mc" << endl;
    return 0;
}
```

2.5. Время выполнение сортировки для каждого объёма.

Таблица 1 - Время выполнение сортировки для каждого объёма.

Количество элементов последовательности	Время выполнения программы(мс)
100	900612
1000	1504522

Вывод

В результате выполнения работы я:

1. Освоил алгоритмы работы с поразрядными операциями и их реализацию на языке программирования C++
2. Реализовать задачу по сортировке данных файла, используя для представления данных файла массив битов