****

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«МИРЭА – Российский технологический университет»**

# РТУ МИРЭА

Отчет по выполнению практического задания 5

**Тема: «РАБОТА С ДАННЫМИ ИЗ ФАЙЛА»**

Дисциплина: Структуры и алгоритмы обработки данных

Выполнил студент Комисарик М.А.

группа ИКБО-20-23

**Москва 2024**

**Цель работы:** освоить приёмы работы с битовым представлением беззнаковых целых чисел, реализовать эффективный алгоритм внешней сортировки на основе битового массива, получить практический опыт по применению алгоритмов поиска в таблицах данных.

Задание 1

Формулировка задачи

Пример – как установить 5-й бит произвольного целого числа в 0 и что получится в результате:

1. unsigned char x = 255; //8-разрядное двоичное число 11111111
2. unsigned char maska = 1; //1=00000001 – 8-разрядная маска
3. x = x & (~(maska << 4)); //результат x=239

1.а. Реализовать вышеприведённый пример, проверьте правильность результата в том числе и на других значениях х.

1.б. Реализовать по аналогии с предыдущим примером установку 7-го бита числа в единицу.

Листинг 1.

1. //Битовые операции
2. #include <cstdlib>
3. #include <iostream>
4. #include <Windows.h>
5. #include <bitset>
6. using namespace std;
7. int main()
8. {
9. SetConsoleCP(1251);
10. SetConsoleOutputCP(1251);
11. unsigned int x = 25;
12. const int n = sizeof(int) \* 8; //=32 - количество разрядов в числе типа int
13. unsigned maska = (1 >> n - 1); //1 в старшем бите 32-рпзярдной сетки
14. cout << "Начальный вид маски: " << bitset<n>(maska) << endl;
15. cout << "Результат: ";
16. for (int i = 1; i <= n; i++) //32 раза - по количеству разрядов
17. {
18. cout << ((x & maska) >> (n - i));
19. maska = maska >> 1; //смещение 1 в маске на разряд вправо
20. }
21. cout << endl;
22. system("pause");
23. return 0;
24. }

1.в. Реализовать код листинга 1, объясните выводимый программой результат.

Математическая модель решения

1.а. В приведенном примере производится побитовое И с переменной x и маской, побитово сдвинутой влево на 4 бита и побитово инвертированной. Таким образом бит маски на 4-й позиции будет равен 0, а все остальные – 1. Из-за этого, после выполнения побитового и с маской и переменной x, 4-й бит переменной x будет установлен в 0, а все остальные не изменятся (x & 0 = 0; x & 1 = x).

1.б. Для установления 7-го бита в 1 требуется применить операцию побитового ИЛИ к маске равной 1, побитого сдвинутой влево на 7 позиций, и переменной x. (x | 0 = x; x | 1 = 1)

1.в. В листинге 1 сначала крайний левый бит маски устанавливается в 1, затем в цикле маска побитово сдвигается вправо и в консоль выводится бит числа x, располагающийся на позиции единицы в маске. Это достигается с помощью побитового И маски и числа x, и побитого сдвига вправо результата до того момента, пока требуемый бит не окажется на младшем разряде. Таким образом происходит вывод двоичного представления числа x.

Код программы

Код программы 1.а:

1. unsigned int inputAmount;
2. unsigned int x;
3. unsigned int mask = 1;
4. cout << "Введите количество вводов x: ";
5. cin >> inputAmount;
6. for (unsigned int i = 0; i < inputAmount; i++)
7. {
8. cout << "Введите x: ";
9. cin >> x;
10. cout << "Установка 4-го бита x в 0\n";
11. x = x & (~(mask << 4));
12. cout << "x = " << x << "\n";
13. }

Код программы 1.б:

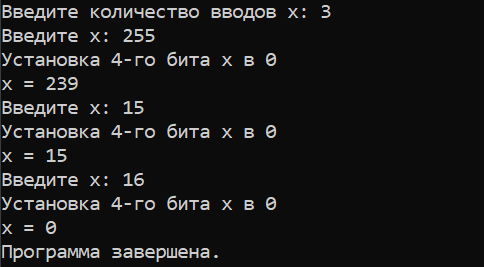
1. unsigned int inputAmount;
2. unsigned int x;
3. unsigned int mask = 1;
4. cout << "Введите количество вводов x: ";
5. cin >> inputAmount;
6. for (unsigned int i = 0; i < inputAmount; i++)
7. {
8. cout << "Введите x: ";
9. cin >> x;
10. cout << "Установка 7-го бита x в 1\n";
11. x = x | mask << 7;
12. cout << "x = " << x << "\n";
13. }

Код программы 1.в:

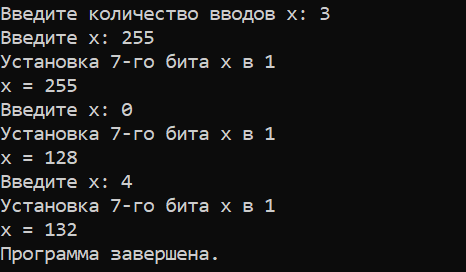
1. unsigned int x = 255;
2. unsigned int mask = 1;
3. const int n = sizeof(int) \* 8;
4. x = 25;
5. mask = 1 << (n - 1);
6. cout << "Начальная маска: " << mask << " = " << bitset<n>(mask) << "\n";
7. cout << "Результат: ";
8. for (unsigned int i = 1; i <= n; i++)
9. {
10. cout << ((x & mask) >> (n - i));
11. mask >>= 1;
12. }
13. cout << "\n";

Результаты тестирования

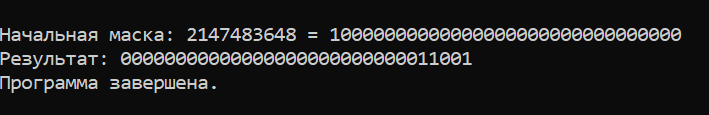
Результаты тестирования программы 1.а на трех входных значениях:

****

Результаты тестирования программы 1.б на трех входных значениях:

****

Результат тестирования программы 1.в:



Задание 2

Формулировка задачи

Пусть даны не более 8 чисел со значениями от 0 до 7, например, {1, 0, 5, 7, 2, 4}.

Подобный набор чисел удобно отразить в виде 8-разрядной битовой последовательности 11101101. В ней единичные биты показывают наличие в исходном наборе числа, равного номеру этого бита в последовательности (нумерация с 0 слева). Т.е. индексы единичных битов в битовом массиве – это и есть числа исходной последовательности.

Последовательное считывание бит этой последовательности и вывод индексов единичных битов позволит естественным образом получить исходный набор чисел в отсортированном виде – {0, 1, 2, 4, 5, 7}.

2.а. Реализовать вышеописанный пример с вводом произвольного набора до 8-ми чисел (со значениями от 0 до 7) и его сортировкой битовым массивом в виде числа типа unsigned char. Проверить работу программы.

2.б. Адаптировать вышеприведённый пример для набора из 64-х чисел (со значениями от 0 до 63) с битовым массивом в виде числа типа unsigned long long.

2.в. Исправить программу задания 2.б, чтобы для сортировки набора из 64-х чисел использовалось не одно число типа unsigned long long, а линейный массив чисел типа unsigned char.

Математическая модель решения

2.а. Для реализации примера необходимо использовать битовые операции сдвига и логическую битовую операцию или. Чтобы представить массив чисел в виде битовой маски, требуется битово сдвинуть единицу влево на k раз для каждого числа k из этого массива и произвести побитовое или с заранее заготовленной пустой (заполненной нулями) маской. Тип переменной маски определяется количеством чисел, которое требуется содержать в этой маске. Для 8 чисел, каждое из которых не больше 7, хватит маски типа char (1 байт, 8 бит).

Если не требуется хранить числа в битовой маске после сортировки, то для доступа к биту маски на позиции n, можно побитово сдвинуть маску на n битов вправо и взять остаток от деления на 2. Таким образом, если производить сдвиг вправо на 1 бит на каждом шагу цикла, можно получить информацию о каждом бите маски. В данной работе для вывода чисел из маски использовался этот метод.

2.б. Для хранения не более 64 чисел, каждое из которых не больше 63, требуется как минимум число unsigned long long (8 байт, 64 бита).

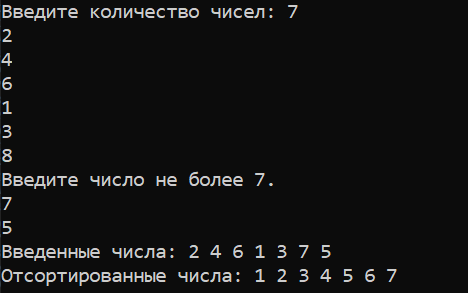
2.в. Чтобы хранить больше чем 64 числа в битовой маске или хранить их не используя типы переменных большие чем один байт, следует использовать массив байтов (char`ов). Чтобы получить доступ к конкретному байту можно обратиться к массиву с индексом равным целочисленному делению требуемого числа на 8, а чтобы получить нужный бит, взять остаток от деления этого числа на 8.

Код программы

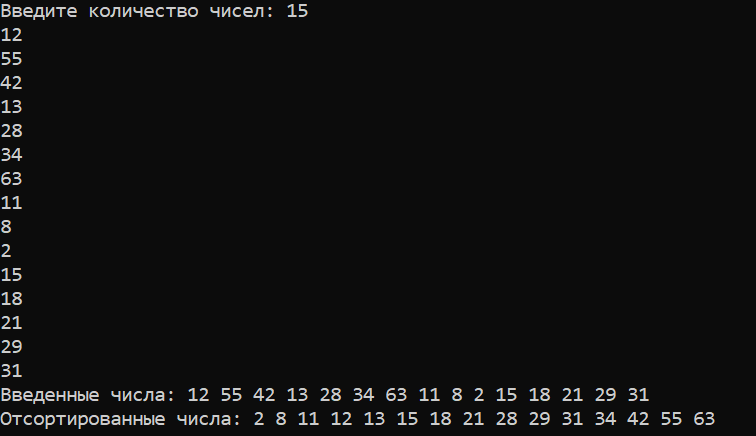
Код программ 2.а, 2.б и 2.в представлен в листингах 2, 3 и 4 соответственно.

Результаты тестирования

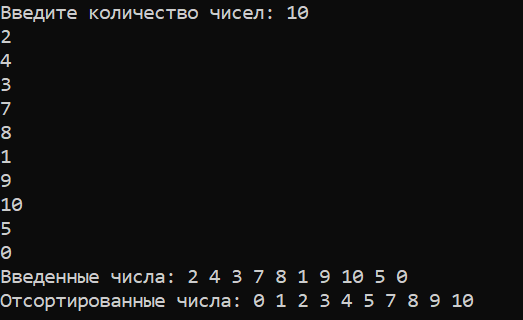
Результат тестирования программы 2.а:



Результат тестирования программы 2.б:



Результат тестирования программы 2.в:



Задание 3

Формулировка задачи.

**ВЫВОДЫ**

Листинг 2.

1. vector<unsigned char> nums;
2. unsigned int inputAmount;
3. cout << "Введите количество чисел: ";
4. do
5. {
6. cin >> inputAmount;
7. if (inputAmount > 8)
8. {
9. cout << "Введите число меньшее 8.\n";
10. }
11. } while (inputAmount > 8);
12. InputNumbers(nums, inputAmount, 8);
13. cout << "Введенные числа: ";
14. OutputNumbers(nums);
15. unsigned char charContainer = 0;
16. for (unsigned char num : nums)
17. {
18. charContainer |= 1 << num;
19. }
20. nums.clear();
21. for (unsigned char i = 0; i < sizeof(char) \* 8; i++)
22. {
23. if (charContainer % 2)
24. {
25. nums.push\_back(i);
26. }
27. charContainer >>= 1;
28. }
29. cout << "Отсортированные числа: ";
30. OutputNumbers(nums);

Листинг 3.

1. vector<unsigned char> nums;
2. unsigned int inputAmount;
3. cout << "Введите количество чисел: ";
4. do
5. {
6. cin >> inputAmount;
7. if (inputAmount > 64)
8. {
9. cout << "Введите число меньшее 64.\n";
10. }
11. } while (inputAmount > 64);
12. InputNumbers(nums, inputAmount, 64);
13. cout << "Введенные числа: ";
14. OutputNumbers(nums);
15. unsigned long long longContainer = 0;
16. for (unsigned char num : nums)
17. {
18. longContainer |= (unsigned long long)1 << num;
19. }
20. nums.clear();
21. for (unsigned int i = 0; i < sizeof(long long) \* 8; i++)
22. {
23. if (longContainer % 2)
24. {
25. nums.push\_back(i);
26. }
27. longContainer >>= 1;
28. }
29. cout << "Отсортированные числа: ";
30. OutputNumbers(nums);

Листинг 4.

1. vector<unsigned char> nums;
2. unsigned int inputAmount;
3. nums.clear();
4. cout << "Введите количество чисел: ";
5. do
6. {
7. cin >> inputAmount;
8. if (inputAmount > 64)
9. {
10. cout << "Введите число меньшее 64.\n";
11. }
12. } while (inputAmount > 64);
13. InputNumbers(nums, inputAmount, 64);
14. cout << "Введенные числа: ";
15. OutputNumbers(nums);
16. vector<unsigned char> charContainers(8);
17. for (unsigned char num : nums)
18. {
19. charContainers[num / 8] |= 1 << num % 8;
20. }
21. nums.clear();
22. for (unsigned int i = 0; i < 64; i++)
23. {
24. if (charContainers[i / 8] % 2)
25. {
26. nums.push\_back(i);
27. }
28. charContainers[i / 8] >>= 1;
29. }
30. cout << "Отсортированные числа: ";
31. OutputNumbers(nums);