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 5.1

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 x.

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:
8:   int main()
9:   {
10:      SetConsoleCP(1251);
11:      SetConsoleOutputCP(1251);
12:
13:      unsigned int x = 25;
14:      const int n = sizeof(int) * 8;      //=32 - количество разрядов в числе
     типа int
15:      unsigned maska = (1 >> n - 1);      //1 в старшем бите 32-рпзярдной сетки
16:      cout << "Начальный вид маски: " << bitset<n>(maska) << endl;
17:      cout << "Результат: ";
18:      for (int i = 1; i <= n; i++)    //32 раза - по количеству разрядов
19:      {
20:          cout << ((x & maska) >> (n - i));
21:          maska = maska >> 1;         //смещение 1 в маске на разряд вправо
22:      }
23:      cout << endl;
24:      system("pause");
25:      return 0;
26:  }
```

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:
5:   cout << "Введите количество вводов x: ";
6:   cin >> inputAmount;
7:
8:   for (unsigned int i = 0; i < inputAmount; i++)
9:   {
10:      cout << "Введите x: ";
11:      cin >> x;
12:      cout << "Установка 4-го бита x в 0\n";
13:
14:      x = x & (~(mask << 4));
15:      cout << "x = " << x << "\n";
16:  }
```

Код программы 1.б:

```
1:   unsigned int inputAmount;
2:   unsigned int x;
3:   unsigned int mask = 1;
4:
5:   cout << "Введите количество вводов x: ";
6:   cin >> inputAmount;
7:
8:   for (unsigned int i = 0; i < inputAmount; i++)
9:   {
10:      cout << "Введите x: ";
11:      cin >> x;
12:      cout << "Установка 7-го бита x в 1\n";
13:
14:      x = x | mask << 7;
15:      cout << "x = " << x << "\n";
16:  }
```

Код программы 1.в:

```
1:   unsigned int x = 255;
2:   unsigned int mask = 1;
3:
4:   const int n = sizeof(int) * 8;
5:   x = 25;
6:   mask = 1 << (n - 1);
7:   cout << "Начальная маска: " << mask << " = " << bitset<n>(mask) << "\n";
8:   cout << "Результат: ";
9:
10:  for (unsigned int i = 1; i <= n; i++)
11:  {
12:      cout << ((x & mask) >> (n - i));
13:      mask >>= 1;
14:  }
15:  cout << "\n";
```

Результаты тестирования

Результаты тестирования программы 1.а на трех входных значениях:

```
Введите количество вводов x: 3
Введите x: 255
Установка 4-го бита x в 0
x = 239
Введите x: 15
Установка 4-го бита x в 0
x = 15
Введите x: 16
Установка 4-го бита x в 0
x = 0
Программа завершена.
```

Результаты тестирования программы 1.б на трех входных значениях:

```
Введите количество вводов x: 3
Введите x: 255
Установка 7-го бита x в 1
x = 255
Введите x: 0
Установка 7-го бита x в 1
x = 128
Введите x: 4
Установка 7-го бита x в 1
x = 132
Программа завершена.
```

Результат тестирования программы 1.в:

```
Начальная маска: 2147483648 = 10000000000000000000000000000000
Результат: 00000000000000000000000011001
Программа завершена.
```

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:

```
Введите количество чисел: 7
2
4
6
1
3
8
Введите число не более 7.
7
5
Введенные числа: 2 4 6 1 3 7 5
Отсортированные числа: 1 2 3 4 5 6 7
```

Результат тестирования программы 2.б:

```
Введите количество чисел: 15
12
55
42
13
28
34
63
11
8
2
15
18
21
29
31
Введенные числа: 12 55 42 13 28 34 63 11 8 2 15 18 21 29 31
Отсортированные числа: 2 8 11 12 13 15 18 21 28 29 31 34 42 55 63
```

Результат тестирования программы 2.в:

```
Введите количество чисел: 10
2
4
3
7
8
1
9
10
5
0
Введенные числа: 2 4 3 7 8 1 9 10 5 0
Отсортированные числа: 0 1 2 3 4 5 7 8 9 10
```

Задание 3

Формулировка задачи

Входные данные: файл, содержащий не более $n=10^7$ неотрицательных целых чисел, среди них нет повторяющихся.

Результат: упорядоченная по возрастанию последовательность исходных чисел в выходном файле.

Время работы программы: ~10 с (до 1 мин. для систем малой вычислительной мощности).

Максимально допустимый объём ОЗУ для хранения данных: 1 МБ.

Очевидно, что размер входных данных гарантированно превысит 1МБ (это, к примеру, максимально допустимый объём стека вызовов, используемого для статических массивов).

Требование по времени накладывает ограничение на количество чтений исходного файла.

3.а. Реализовать задачу сортировки числового файла с заданными условиями. Добавить в код возможность определения времени работы программы.

3.б. Определить программно объём оперативной памяти, занимаемый битовым массивом.

Математическая модель решения

Для сортировки используется тот же принцип, что и во втором задании. Для этого требуется хранить битовую маску в оперативной памяти. Числа в файле записаны построчно, поэтому для оптимизации счета чисел из файла используется функции getline() и stoi(). Чтобы числа в файле не повторялись перед сортировкой происходит запись всех чисел от 0 до n в обратном порядке, где n указывается в консоли. Для оптимизации записи чисел в файл после сортировки был использован буфер размером 1Кб и метод write(), принимающий массив char`ов. Максимальное количество оперативной памяти, которое можно использовать в задании равно 1Мб, что равно 1024 * 1024 * 8 = 8388608 бит, такое количество чисел можно записать в битовую маску. В задании было использовано 8300000 чисел, чтобы можно было хранить буфер и дополнительные данные не выходя за рамки 1Мб.

Код программы

Код программы 3.а представлен в листинге 5.

Код программы 3.б идентичен коду программы 3.а, но в код добавлены следующие строчки:

```
1:    unsigned int memCount = 0;
2:    memCount += cBufferSize;
3:    memCount += numsAmount / 8;
4:    cout << "Размер использованной оперативной памяти: " << memCount / 1024.0 /
      1024.0 << " Mb\n";
```

Результаты тестирования

Результаты тестирования программы 3.б:

```
Задание 3.б
Введите количество чисел: 8300000
Числа записаны в файл. Для продолжения нажмите любую клавишу . . .
Затраченное время: 12.439 с.
Размер использованной оперативной памяти: 0.990414 Mb
```

Результаты тестирования программы 3.а аналогичны, за исключением отсутствия счетчика памяти.

ВЫВОДЫ

Битовая сортировка является очень быстрым способом сортировки неповторяющихся неотрицательных целых чисел, однако имеет недостаток – необходимость хранить все числа в оперативной памяти при сортировке.

В данной работе были получены навыки работы с файлами и битовым представлением целых неотрицательных чисел.

Источники

Листинг 2.

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

Листинг 3.

```
1:   vector<unsigned char> nums;
2:   unsigned int inputAmount;
3:
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:
14:  InputNumbers(nums, inputAmount, 64);
15:  cout << "Введенные числа: ";
16:  OutputNumbers(nums);
17:
18:  unsigned long long longContainer = 0;
19:  for (unsigned char num : nums)
20:  {
21:      longContainer |= (unsigned long long)1 << num;
22:  }
23:  nums.clear();
24:  for (unsigned int i = 0; i < sizeof(long long) * 8; i++)
```

```
25:  {
26:      if (longContainer % 2)
27:      {
28:          nums.push_back(i);
29:      }
30:      longContainer >>= 1;
31:  }
32:  cout << "Отсортированные числа: ";
33:  OutputNumbers(nums);
```

Листинг 4.

```
1:   vector<unsigned char> nums;
2:   unsigned int inputAmount;
3:
4:   nums.clear();
5:   cout << "Введите количество чисел: ";
6:   do
7:   {
8:       cin >> inputAmount;
9:       if (inputAmount > 64)
10:      {
11:          cout << "Введите число меньшее 64.\n";
12:      }
13:  } while (inputAmount > 64);
14:
15:  InputNumbers(nums, inputAmount, 64);
16:  cout << "Введенные числа: ";
17:  OutputNumbers(nums);
18:
19:  vector<unsigned char> charContainers(8);
20:  for (unsigned char num : nums)
21:  {
22:      charContainers[num / 8] |= 1 << num % 8;
23:  }
24:  nums.clear();
25:  for (unsigned int i = 0; i < 64; i++)
26:  {
27:      if (charContainers[i / 8] % 2)
28:      {
29:          nums.push_back(i);
30:      }
31:      charContainers[i / 8] >>= 1;
32:  }
33:  cout << "Отсортированные числа: ";
34:  OutputNumbers(nums);
```

Листинг 5.

```
1:   ofstream initFile("nums.txt", ios_base::trunc | ios_base::out);
2:   unsigned int numsAmount;
3:   cout << "Введите количество чисел: ";
4:   cin >> numsAmount;
5:
6:   const int cBufferSize = 1024;
7:   char cBuffer[cBufferSize];
8:
9:   unsigned int count = 0;
10:
11:  for (unsigned int i = 0; i < numsAmount; ++i)
12:  {
13:      unsigned int rI = numsAmount - i - 1;
14:      _itoa(rI, cBuffer + count, 10);
```

```
15:       count += rI != 0 ? log10(rI) + 2 : 2;
16:       cBuffer[count - 1] = '\n';
17:       if (count > cBufferSize - 8)
18:       {
19:           initFile.write(cBuffer, count);
20:           count = 0;
21:       }
22:   }
23:   if (count > 0)
24:   {
25:       initFile.write(cBuffer, count);
26:       count = 0;
27:   }
28:
29:   initFile.close();
30:
31:   cout << "Числа записаны в файл. ";
32:   system("pause");
33:
34:   auto start = chrono::high_resolution_clock::now();
35:
36:   ifstream inFile("nums.txt", ios_base::in);
37:   vector<unsigned char> charContainers(numsAmount / 8, 0);
38:   unsigned int buffer;
39:   string sBuffer;
40:
41:   for (unsigned int i = 0; i < numsAmount; ++i)
42:   {
43:       getline(inFile, sBuffer);
44:       buffer = stoi(sBuffer);
45:       charContainers[buffer / 8] |= 1 << (buffer % 8);
46:   }
47:
48:   inFile.close();
49:
50:   ofstream wFile("nums.txt", ios_base::trunc | ios_base::out);
51:
52:   for (unsigned int i = 0; i < numsAmount; i++)
53:   {
54:       if (charContainers[i / 8] % 2)
55:       {
56:           _itoa(i, cBuffer + count, 10);
57:           count += i != 0 ? log10(i) + 2 : 2;
58:           cBuffer[count - 1] = '\n';
59:           if (count > cBufferSize - 8)
60:           {
61:               wFile.write(cBuffer, count);
62:               count = 0;
63:           }
64:       }
65:       charContainers[i / 8] >>= 1;
66:   }
67:   if (count > 0)
68:   {
69:       wFile.write(cBuffer, count);
70:   }
71:
72:   wFile.close();
73:
74:   auto end = chrono::high_resolution_clock::now();
75:   auto duration = end - start;
76:
77:   cout << "Затраченное время: " <<
      (chrono::duration_cast<chrono::milliseconds>(duration)).count() / 1000.0 << "
      c.\n";
```