МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«МИРЭА – Российский технологический университет»

РТУ МИРЭА

ОТЧЕТ
ПО ПРАКТИЧЕСКОЙ РАБОТЕ №1

Сортировка числового файла с помощью битового массива
по дисциплине
«Структуры и алгоритмы обработки данных»

Выполнил студент группы ИКБО-01-21 Боронин Н.А

Практическая работа выполнена «__» сентября 2022 г. _____
(*подпись студента*)

«Зачтено» «__» _____2022 г. _____
(подпись руководителя)

Москва 2022

1. Цель работы

Целью данной работы является освоение приёмов работы с битовым представлением беззнаковых целых чисел, и реализация эффективного алгоритма внешней сортировки на основе битового массива.

2. Ход работы

2.1. Задание 1

a Формулировка задачи

Необходимо реализовать следующие программы:

- приведенный в методических указаниях пример, проверить его правильность на других значениях переменной x;
- установка 7-го бита числа в единицу;
- приведенный в листинге 1 пример, объяснив выводимый программой результат.

b Описание алгоритма

Касаемо второй задачи: для того, чтобы установить 7-ой бит числа в единицу, необходимо применить побитовый сдвиг влево на шесть разрядов.

Третья программа работает таким образом: маской является 32-х разрядное число с единицей в старшем разряде. Далее в цикле на каждом i-том шаге после выполнения побитового И, производится побитовый сдвиг вправо таким образом, чтобы в младшем разряде оказался i-тый бит числа и выводится результат. После этого единица в маске смещается на один разряд вправо, таким образом, чтобы можно было вывести следующий бит числа.

c Код программы

Ниже приведен код программ, решающий указанные выше задачи.

```
#include <bits/stdc++.h>
using namespace std;
int main() {
    unsigned x = 255;
    unsigned char maska = 1;
    x = x & (~(maska << 4));
    // maska = 00000001
    // maska << 4 = 00010000
    //~() = 11101111
    //128 + 64 + 32
    cout << x;
    system("pause");
}
```

Рисунок 1 – код программы к заданию 1.a

```cpp
#include<bits/stdc++.h>
using namespace std;
int main() {
    unsigned int x = 1; //00000001
    unsigned char maska = 1; //00000001
    x = x | maska << 6; //001000001
    cout << x;
    system("pause");
}
```

Рисунок 2 – код программы к заданию 1.б

```cpp
#include<bits/stdc++.h>
using namespace std;
int main() {
    unsigned int x = 25;
    const int n = sizeof(int)*8; // получили 32 нуля подряд
    unsigned maska = (1 << n - 1); // получили 1 и 31 ноль
    cout << "Mask: " << bitset<n>(maska) << endl;
    cout << "Result: ";
    for(int i = 1; i <= n; i++) {
        cout << ((x & maska) >> (n - i)); //выводим i-тый старший бит числа
        maska = maska >> 1; //уменьшаем рабочую длину маски
    }
    cout << endl;
    system("pause");
}
```

Рисунок 3 – код программы к заданию 1.в

d Результаты тестирования

Протестируем написанные программы, чтобы убедиться в их работоспособности.

```
239
Для продолжения нажмите любую клавишу
```

Рисунок 4 – результат работы программы к заданию 1.а

```
65
Для продолжения нажмите любую клавишу
```

Рисунок 5 – результат работы программы к заданию 1.б

```
Mask: 10000000000000000000000000000000
Result: 00000000000000000000000000011001
```

Рисунок 6 – результат работы программы к заданию 1.в

Результаты тестирования соответствуют теоретическим вычислениям, следовательно код корректно решает поставленные задачи.

2.2. Задание 2

a Формулировка задачи

Используя сортировку при помощи битового массива, нужно последовательно решить три задачи:

- ввод произвольного набора до 8-ми чисел (со значениями [0, 7]) и его сортировка битовым массивом в виде числа unsigned char;
- адаптировать предыдущую программу для набора до 64-х чисел с битовым массивом типа unsigned long long;
- отсортировать массив, содержащий до 64-х значений, когда в качестве битового массива используется линейный массив чисел типа unsigned char.

b Описание алгоритма

Для того, чтобы отсортировать массив, отобразим его в виде последовательности нулей и единиц, где 1 в i-том разряде – означает присутствие числа i в массиве. После этого последовательно считаем биты этой последовательности и выведем индексы тех разрядов, где стоит единица.

Как определять, с каким элементом битового массива надо взаимодействовать, когда в качестве него нужно будет линейный массив типа unsigned char? Целочисленное деление на количество бит в unsigned char (на 8) даст номер элемента битового массива, а результат от деления по модулю 8 укажет конкретный бит, который и будет представлять данное число.

В целях оптимизации, можно добавлять новые блоки битового массива постепенно. Как только встречается число, которое нельзя записать в уже существующую разрядную сетку, добавим в цикле столько элементов типа unsigned char, сколько необходимо.

c Код программы

Реализуем описанный выше алгоритм.

```cpp
#include<bits/stdc++.h>
using namespace std;
int main() {

    const int kBitGrid = 8; //длина разрядной сетки
    int shift = kBitGrid - 1;
    unsigned char mask = 0;
    int n = 0;

    cin >> n;
    vector<int>nums = vector<int>(n, 0);

    for(int i = 0; i < n; i++) {
        cin >> nums[i];
        // битовый массив имеет следующий вид
        // 0 1 0 0 1 0 1 1
        // 0 1 2 3 4 5 6 7
        //из этих соображений вычисляем необходимую длину свдига
        mask = mask | 1 << shift - nums[i];
    }

    for(int i = 1; i < kBitGrid; i++)
        //если i-тый бит - единица, выводим его
        if (mask >> (shift - i) & 1) cout << i << " ";
    system("pause");
}
```

Рисунок 7 – код программы к заданию 2.а

```cpp
#include<bits/stdc++.h>
using namespace std;
int main() {

    const int kBitGrid = 64;
    int shift = kBitGrid - 1;
    unsigned long long mask = 0;
    int n = 0;

    cin >> n;
    vector<int>nums = vector<int>(n, 0);

    for(int i = 0; i < n; i++) {
        //идея работы такая же, как в предыдущей задачи
        //единственное, что запишем литерал типа unsigned long long int
        cin >> nums[i];
        mask = mask | 1ull << (shift - nums[i]);
    }

    for(int i = 0; i < kBitGrid; i++)
        if ((mask >> shift - i) & 1) cout << i << " ";
        //if(mask >> i & 1) cout << shift - i << " "; //сортировка по убыванию
    system("pause");
}
```

Рисунок 8 – код программы к заданию 2.б

```cpp
#include<bits/stdc++.h>
using namespace std;

int main() {
    const int kshift = 7;
    unsigned char block = 0;
    int n = 0;

    cin >> n;
    vector<int> nums = vector<int>(n, 0);
    vector<unsigned char> mask = vector<unsigned char>(0);

    for(int i = 0; i < n; i++) {
        cin >> nums[i];
        int block_n = nums[i] / 8; //номер блока, в который попадет число
        int sz = mask.size() - 1;
        if (block_n > sz) {
            ///добавили необходимое количество новых блоков
            for(int j = 0; j < block_n - sz; j++) mask.push_back(block);
        }
        //обновили маску в блоке по анлалогиии с предыдущим заданием
        mask[block_n] = mask[block_n] | 1 << (kshift - nums[i] % 8);
    }

    //процесс вывода аналогичен
    for(int i = 0; i < mask.size() * 8; i++) {
        if ((mask[i / 8] >> (kshift - i % 8)) & 1) cout << i << " ";
    }
    system("pause");
}
```

Рисунок 9 – код программы к заданию 2.в

d Результаты тестирования

Протестируем написанные программы:

```
4
7 5 3 1
1 3 5 7
Для продолжения нажмите любую клавишу . . .
```

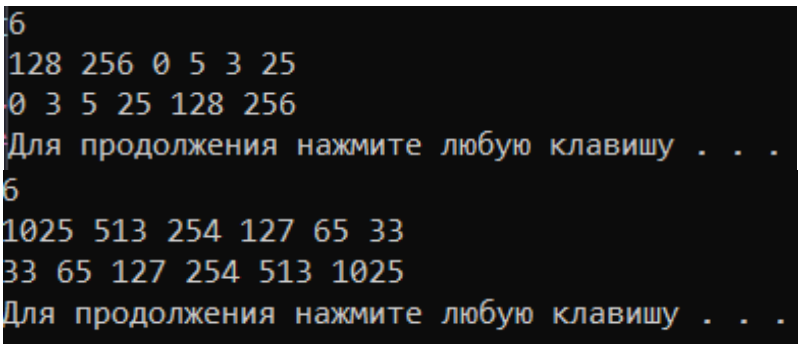
```
7
1 3 5 7 6 4 2
1 2 3 4 5 6 7
Для продолжения нажмите любую клавишу . . .
```

Рисунки 10-11 – результаты работы программы к заданию 2.а

```
6
63 0 35 56 3 13
0 3 13 35 56 63
Для продолжения нажмите любую клавишу . . .
```

```
5
30 50 63 25 0
0 25 30 50 63
Для продолжения нажмите любую клавишу . . .
```

Рисунки 12-13 – результаты работы программы к заданию 2.б



Рисунки 14-15 – результаты работы программы к заданию 2.в

Тестирование показало, что программы работают корректно, а код к заданию 2.в может сортировать любые значения, не привязываясь к разрядности сетки, при условии, что значения встречаются только один раз

2.3.Задание 3

a Формулировка задачи

Используя битовый массив реализовать высокоэффективную сортировку большого объема числовых данных в массиве. Необходимо учитывать следующие ограничения:

- время работы – 10с;
- максимальный объем используемой памяти – 1МБ.

Кроме того, требуется замерить время работы и объем занятой памяти.

b Описание алгоритма

Генерация случайных уникальных чисел будет реализована так:

- создается vector, выбранного пользователем размера n
- при помощи функции iota он заполняется числами от 0 до n
- при помощи функции shuffle элементы массива перемешиваются случайным образом

Сам алгоритм сортировки работает аналогично тому, что использовался в задании 2.

c Код программы

```cpp
#include<bits/stdc++.h>
using namespace std;
void create_file() {
    int n;
    cout << "Amount of numbers: ";
    cin >> n;

    ofstream out;
    out.open("input.txt");

    auto rng = std::default_random_engine {}; //способ генерации случайного числа
    vector<int> nums = vector<int>(n);
    iota(nums.begin(), nums.end(), 0);
    shuffle(nums.begin(), nums.end(), rng); //перетасуем массив

    for(int i = 0; i < n; i++) {
        out.width(log10(n) + 1); //ширина вывода - число знаков в самом большом числе
        out << nums[i] << " ";
        if (!(i % 10)) out << "\n";
    }
    out.close();
}

int main() {
    create_file();

    auto start_time = chrono::steady_clock::now();

    ifstream in;
    in.open("input.txt");

    const int kshift = 7;
    unsigned char block = 0;
    int num = 0;
    vector<unsigned char> mask = vector<unsigned char>(0);

    while(in >> num) {
        int block_n = num / 8;
        int sz = mask.size() - 1;
        if (block_n > sz) {
            for(int j = 0; j < block_n - sz; j++) mask.push_back(block);
        }
        mask[block_n] = mask[block_n] | 1 << (kshift - num % 8);
    }
    in.close();
```

```
    ofstream out;
    out.open("output.txt");

    int len = mask.size() * 8;
    for(int i = 0; i < len; i++) {
        out.width(log10(len) + 1);
        if ((mask[i / 8] >> (kshift - i % 8)) & 1) out << i << " ";
        if(!(i % 20) && i) out << "\n";
    }
    out.close();

    auto end_time = chrono::steady_clock::now();
    auto time = chrono::duration_cast<chrono:: milliseconds>(end_time - start_time);
    //int kbits = (mask.size() * sizeof(unsigned char)) / 1024;

    cout << "Time spent: " << time.count() << "ms" << "\n";
    //cout << "Array takes up " << kbits << " kb of RAM" << "\n";

    system("pause");
}
```

Рисунки 16-18 – код программы к заданию 3.а и 3.б

d Результаты тестирования

Протестируем программу на наибольшем количестве чисел – 10^7

```
Amount of numbers: 10000000
Time spent: 6523ms
Для продолжения нажмите любую клавишу . . .
```

1	551205	484960	855745	392631	883232	375922	676155	340688	842897	99428
2	497066	105526	721046	177120	422046	977827	290163	409342	721216	549879
3	957169	404099	946466	595287	121525	951282	137224	744107	944705	723587
4	756044	531667	773219	366384	210837	408371	766790	296309	903407	601572
5	995638	608408	927424	741341	973537	629306	323176	813258	534677	805297
6	275643	421036	327366	919115	440663	44871	344032	415991	13536	109915
7	470181	594256	253700	540673	396830	662766	405168	804389	130151	713024
8	740996	471898	222730	658606	925096	369088	945818	725792	599078	134587
9	866264	342208	813874	810215	292960	101238	878421	642384	511248	377805
10	885814	609696	482048	49456	618333	478675	372924	493506	944005	650290
11	657523	412347	630403	214119	291943	769135	530796	444808	711884	42861
12	469899	611721	338979	661190	392029	494347	454420	540549	525692	319213
13	758508	512656	782861	10691	434850	895582	787497	844985	520750	618184
14	296784	389003	582987	776352	798754	675515	946925	610522	485155	993881
15	962708	842329	423082	871874	637759	693931	179465	486255	245091	320638
16	812137	82223	942069	939142	829577	447830	539717	239399	670594	878889
17	604119	353641	907533	919005	634321	311355	720619	995662	209181	467221
18	451140	25139	281566	648510	571214	989694	288747	187040	51746	80662
19	495300	945022	19339	860671	662505	258748	737579	311848	176931	20510
20	843046	365607	303293	705649	735565	840161	974126	109514	453397	338025
21	118687	561518	9808	472325	352012	242248	281120	472283	683116	643869
22	858629	843389	335775	273953	216230	137105	949362	166634	394181	907506
23	591789	784218	453907	99114	834912	541298	825969	378864	346375	62224
24	683569	194980	589243	734237	413674	19941	980805	834712	775975	387450
25	15918	229532	24558	872389	680666	385212	781143	25361	488163	363592
26	614186	530879	508237	633079	131995	777697	525668	702917	653343	683076
27	411294	397277	327707	564208	233693	156572	232086	954908	242978	275253
28	427646	746981	676471	338883	6423	627143	531446	641473	308978	74063
29	782175	613656	169631	873293	196746	234761	666966	512690	271072	440798
30	302570	327515	990551	585622	686745	845987	682842	533350	135245	624841
31	585326	149352	883582	522744	53344	376389	274864	528324	932455	558869

	0	1	2	3	4	5	6	7	8	9
2	10	11	12	13	14	15	16	17	18	19
3	20	21	22	23	24	25	26	27	28	29
4	30	31	32	33	34	35	36	37	38	39
5	40	41	42	43	44	45	46	47	48	49
6	50	51	52	53	54	55	56	57	58	59
7	60	61	62	63	64	65	66	67	68	69
8	70	71	72	73	74	75	76	77	78	79
9	80	81	82	83	84	85	86	87	88	89
10	90	91	92	93	94	95	96	97	98	99
11	100	101	102	103	104	105	106	107	108	109
12	110	111	112	113	114	115	116	117	118	119
13	120	121	122	123	124	125	126	127	128	129
14	130	131	132	133	134	135	136	137	138	139
15	140	141	142	143	144	145	146	147	148	149
16	150	151	152	153	154	155	156	157	158	159
17	160	161	162	163	164	165	166	167	168	169
18	170	171	172	173	174	175	176	177	178	179
19	180	181	182	183	184	185	186	187	188	189
20	190	191	192	193	194	195	196	197	198	199
21	200	201	202	203	204	205	206	207	208	209
22	210	211	212	213	214	215	216	217	218	219
23	220	221	222	223	224	225	226	227	228	229
24	230	231	232	233	234	235	236	237	238	239
25	240	241	242	243	244	245	246	247	248	249
26	250	251	252	253	254	255	256	257	258	259
27	260	261	262	263	264	265	266	267	268	269
28	270	271	272	273	274	275	276	277	278	279
29	280	281	282	283	284	285	286	287	288	289
30	290	291	292	293	294	295	296	297	298	299
31	300	301	302	303	304	305	306	307	308	309

Рисунки 19-21 – результаты работы программы к заданию 3.а(время работы, фрагмент входного файла, фрагмент выходного файла)

Используя программные средства, выведем объем памяти, занимаемый при сортировке 10^7 чисел.

```
Amount of numbers: 10000000
Time spent: 6017ms
Array takes up 1220 kb of RAM
Для продолжения нажмите любую клавишу . . .
```

Рисунок 22 – часть результата работы программы к заданию 3.б

Получается, что программа может отсортировать чуть меньше 9×10^6 чисел, чтобы объем занимаемой памяти не превышал 1 МБ:

```
Amount of numbers: 9000000
Time spent: 5543ms
Array takes up 1098 kb of RAM
Для продолжения нажмите любую клавишу . . .
```

Рисунок 23 – входные данные, удовлетворяющие ограничениям

3. Вывод

В ходе работы были получены знания о приёмах работы с битовым представлением беззнаковых целых чисел, а также были реализованы несколько сортировок, в том числе эффективный алгоритм внешней сортировки на основе битового массива. Следовательно, можно говорить о достижении поставленной цели