Национальный исследовательский университет
"Московский авиационный институт"
Факультет No8 "Информационные технологии и прикладная математика"
Кафедра 806 "Вычислительная математика и программирование"

# ЛАБОРАТОРНАЯ РАБОТА №1
# ПО КУРСУ “ДИСКРЕТНЫЙ АНАЛИЗ”
# 3 СЕМЕСТР

Выполнил студент: Поляков А.И.
Группа: М80-208Б-19
Оценка:
Подпись:

Москва 2020

# Лабораторная работа №1

**Вариант №1-1**

Требуется разработать программу, осуществляющую ввод пар «ключ-значение», их упорядочивание по возрастанию ключа указанным алгоритмом сортировки за линейное время и вывод отсортированной последовательности.

Вариант задания определяется типом ключа (и соответствующим ему методом сортировки) и типом значения:

**Сортировка подсчётом.**

*Тип ключа:* числа от 0 до 65535.

*Тип значения:* строки фиксированной длины 64 символа, во входных данных могут встретиться строки меньшей длины, при этом строка дополняется до 64-х нулевыми символами, которые не выводятся на экран.

# Описание

Требуется описать алгоритм сортировки подсчетом. В качестве входных значений выступают пары ключ - значение. Ключом являются числа от 0 до 65535.

**Сортировка подсчетом** — простейший способ упорядочить массив за линейное время. Применять его можно только для целых чисел, небольшого диапазона, т.к. он требует *O(M)* дополнительной памяти, где M - ширина диапазона сортируемых чисел. Алгоритм особо эффективен когда мы сортируем большое количество чисел, значения которых имеют небольшой разброс.

Идея этого алгоритма заключается в том, чтобы для каждого элемента x предварительно подсчитать, сколько элементов входной последовательности меньше x. Далее x записывается напрямую в выходной массив в соответствии с этим числом (если, скажем, 5 элементов входного массива меньше x, то в выходном массиве x должен быть записан на место номер 6). Если предположить, что в сортируемой последовательности могут присутствовать равные числа, то представленная схема потребует небольшой модификации, позволяющей избежать записи нескольких чисел на одно место.

# Исходный код

На каждой непустой строке входного файла располагается пара «ключ-значение», поэтому создадим новую структуру *NSort :: TPair*, в которой будем хранить ключ и значение.

| **sort.hpp** | |
|---|---|
| void counting_sort( NVector::TVector< NMytypes::TPair<int,char[65]» & elems, unsigned $\max_e lem$); | Функция сортировки подсчетом |
| **pair.hpp** | |
| void print_pair() | Функция для вывода пары на экран |
| void set_pair() | Сеттер для пары |
| **vector.hpp** | |
| TVector() | Конструктор по умолчанию |
| TVector(size_t size) | Конструктор, задающий размер вектора |
| TVector(size_t size, const T & initial) | Конструктор, который создаёт вектор определенного размера, заполненный элементами переданного значения |
| TVector(const TVector<T> & vector) | Конструктор, создающий вектор из другого вектора |
| TVector() | Деструктор |
| T* Begin() | Указатель на начало вектора |
| T* End() | Указатель на конец вектора |
| size_t Capacity() const | Метод получения размера буфера вектора |
| size_t Size() & const | Метод получения размер заполненной части вектора |
| bool Empty() & const | Метод проверки вектора на пустоту |
| void PushBack(const T & value) | Метод добавление элемента в конец вектора |
| T & PopBack() | Метод удаление последнего элемента с конца вектора, возвращает его значение |
| T & operator[](size_t idx) | Метод изменение элемента на заданной позиции |
| T operator[](size_t idx) const | Метод получения значения элемента на заданной позиции |
| void Copy(TVector<T> & vector) | Метод, копирующий вектор, в текущий вектор |

```
namespace NMytypes {
        template<class T, class U >
        struct TPair
        {
                T key;
                U value;
                TPair(): key(0), value("") {}
                TPair(T fir, U sec): key(fir), value(sec) {}
                ~TPair() {}

                void print_pair() {
                        printf("%d %s\n", this->key, this->value);
                }
                void set_pair(T key, U val){
                        this->key = key;
                        strcpy(this->value, val);
                }
        };
}
```

```
namespace NVector {
    template<class T>
    class TVector{
    public:
        TVector();
        TVector(size_t size);
        TVector(size_t size, const T & initial);
        TVector(const TVector<T> & vector);
        ~TVector();

        T* Begin();
        T* End();

        size_t Capacity() const;
        size_t Size() const;
        bool Empty() const;

        void PushBack(const T & value);
        T PopBack();
        T & operator[](size_t idx);
        T operator[](size_t idx) const;
        void Copy(TVector<T> & vector);

    private:
        size_t bufferSize;
        size_t bufferUsed;
        T *buffer;
    };
```

# Консоль

```
user@AN-LAP-1110:/mnt/d/_ДОЛГИ_/да/DA_Labs/lab1$ cat test.t
20 licuieasiabscrhtikmxtarerormggasnyqyppnzqlbndazzufphgjihhgckpnhw
28288 dnxshxyjdaouoyfcfxgzfhxpbpvtbqnhnlvrnsrhkowkyzhgzqvdylzqqxezogzg
25604 xmnycywlwfoovmpnggnqzshrwmkjhfjkcaydlvpinkdlbvzjpobcffuncuiamfmo
53920 hguuvtlajlhmdmxvdbetbrwgktyfodrhzlmkeawowmvvcwolkscgtqlkwhgnvydt
62238 gpdbtpgxescgromuldphkwqtuqblxlwxjbclydiwcffkmmyasxrsffzrdcdtgtzx
25139 ubqzdckhlgrbeojyzesagajivsfrnbaftqulzjbnkikjavnvjhklfkpihentfzhf
55157 pwyflaowvoblqgkuipruifurymtwgxrqlpwbwriyutwiunxxzatulnpvsogqusgg
36971 jmtxsmcngycpkhgbazirbwdciwonwwnyotgwjfovhipllzuyieyibbjitwxsgjtj
18 cgflkincgynssgpqdaelvlanwtpcdvpzeblygrroqlkpjtksdzdmfgtgouygvfby
22861 gedfkfsdmbnfyjldiyijsjkzpcooavfpggpvvdwdvisieihjlhjqjtqbstglzapf
user@AN-LAP-1110:/mnt/d/_ДОЛГИ_/да/DA_Labs/lab1$ ./runner.sh
rm -f *.o
g++ -std=c++14 -pedantic -Wall -c main.cpp -o main.o
g++ -std=c++14 -pedantic -Wall -c sort.cpp -o sort.o
g++ -std=c++14 -pedantic -Wall main.o sort.o -o solution
Running ...

18 cgflkincgynssgpqdaelvlanwtpcdvpzeblygrroqlkpjtksdzdmfgtgouygvfby
20 licuieasiabscrhtikmxtarerormggasnyqyppnzqlbndazzufphgjihhgckpnhw
22861 gedfkfsdmbnfyjldiyijsjkzpcooavfpggpvvdwdvisieihjlhjqjtqbstglzapf
25139 ubqzdckhlgrbeojyzesagajivsfrnbaftqulzjbnkikjavnvjhklfkpihentfzhf
25604 xmnycywlwfoovmpnggnqzshrwmkjhfjkcaydlvpinkdlbvzjpobcffuncuiamfmo
28288 dnxshxyjdaouoyfcfxgzfhxpbpvtbqnhnlvrnsrhkowkyzhgzqvdylzqqxezogzg
36971 jmtxsmcngycpkhgbazirbwdciwonwwnyotgwjfovhipllzuyieyibbjitwxsgjtj
53920 hguuvtlajlhmdmxvdbetbrwgktyfodrhzlmkeawowmvvcwolkscgtqlkwhgnvydt
55157 pwyflaowvoblqgkuipruifurymtwgxrqlpwbwriyutwiunxxzatulnpvsogqusgg
62238 gpdbtpgxescgromuldphkwqtuqblxlwxjbclydiwcffkmmyasxrsffzrdcdtgtzx
```

# Вывод

При выполнении лабораторной работы по "Дискретному анализу"я научился работать с сортировкой подсчетом. Пригодились навыки олимпиадного программирования, связанные с ускорением ввода данных. Ограничился выключением синхронизации стандартных потоков C и C++.

В ходе тестирование программы овладел базовыми навыками программирования на Python, для редактирования генератора тестов. Также научился тестировать скорость выполнения программы с помощью std::chrono и сравнивать её со встроенными алгоритмами сортировок.

Научился тому, как можно использовать свой контейнер в std сортировках. Для себя сделал выводы, что необходимо сразу структурировать свою программу на отдельные файлы, чтобы потом было легче переписывать код. Также, в самом начале, необходимо основательно обдумывать все производимые действия и оценивать сложность их выполнения, это позволит на конечно этапе избежать поисков путей ускорения программы.