

Содержание

Теория	2
Структуры	2
Сортировка «пузырьком»	2
Задание	3
Блок-схемы алгоритмов	4
Функция «readPlanesFromFile»	5
Функция «ErrorHandler»	6
Функция «BubbleSort»	7
Код программы.....	8
Laba3.cpp	8
Тестирование программы.....	12
1. Тестирование некорректных режимов.....	12
2. Тестирование корректных режимов.....	13
Вывод	14

Теория

Структуры

Структуры в C++ — это пользовательские типы данных, которые могут содержать различные переменные разных типов внутри себя. Они позволяют объединять данные, относящиеся к одному объекту или концепции в единый блок. Структуры могут содержать поля и функции-члены, которые могут работать с этими полями.

Сортировка «пузырьком»

Сортировка пузырьком — это простой алгоритм сортировки, который последовательно сравнивает и меняет местами соседние элементы в массиве до тех пор, пока все элементы не будут упорядочены.

Принцип работы сортировки пузырьком:

1. Начинается сравнение первого и второго элементов массива. Если первый элемент больше второго, то они меняются местами.
2. Затем сравниваются второй и третий элементы, и так далее, до конца массива. Если в результате сравнения элементы меняются местами, то считается, что была выполнена одна итерация.
3. После первой итерации на последнем месте окажется наибольший элемент массива.
4. Процесс повторяется для оставшихся элементов массива (от первого до $n-1$), пока все элементы не будут упорядочены.

Пример кода:

```
void bubbleSort(int arr[], int size) {  
    for (int i = 0; i < size - 1; i++) {  
        // Внутренний цикл для сравнения и перестановки соседних элементов  
        for (int j = 0; j < size - i - 1; j++) {  
            // Сравниваем два соседних элемента  
            if (arr[j] > arr[j + 1]) {  
                // Если левый элемент больше правого, меняем их местами  
                int temp = arr[j];  
                arr[j] = arr[j + 1];  
                arr[j + 1] = temp;  
            }  
        }  
    }  
}
```

Задание

ВАРИАНТ № 10

В процессе функционирования АСУ ВД в файле фиксируются данные о самолетах, совершивших посадку. Каждая запись имеет структуру типа:

473	ТУ-154М	Б-3726	3
номер	марка ЛА	бортовой	пункт
рейса		номер	прибытия

1) подготовить программу, сортирующую записи с использованием индексной сортировки методом «пузырька» в порядке возрастания номеров рейсов; результаты печатать в виде таблицы;

2) обеспечить входной контроль бортового номера, номера рейса и пункта прибытия, выполнить отладку и тестирование.

Чтение данных их файла производить с использованием функций ввода/вывода языка C++.

Алгоритм должен быть параметризован; обмен данными с подпрограммой должен осуществляться только через параметры; исходные данные хранятся в отдельном файле.

Блок-схемы алгоритмов

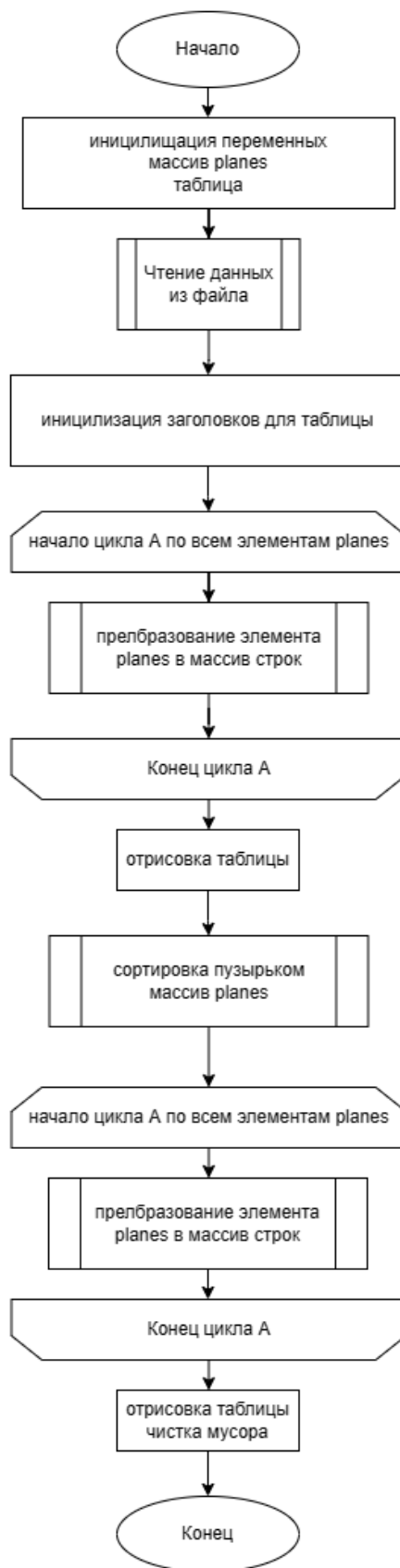


Рис. 1 - Блок-схема функции "laba3"

Функция «readPlanesFromFile»

1. Назначение

Считывание данных из файла и заполнение массива записей полетов

2. Прототип функции:

```
char readPlanesFromFile(Plane*& planes, int& size)
```

3. Обращение:

```
ret = readPlanesFromFile(planes, size);
```

4. Описание параметров

Идентификатор	Тип	Назначение	Входной\выходной
planes	Plane*&	Ссылка на цказатель самолетов	Входной
size	int&	Сылка на размер массива	Входной
result	char	Результат работы функции	выходной

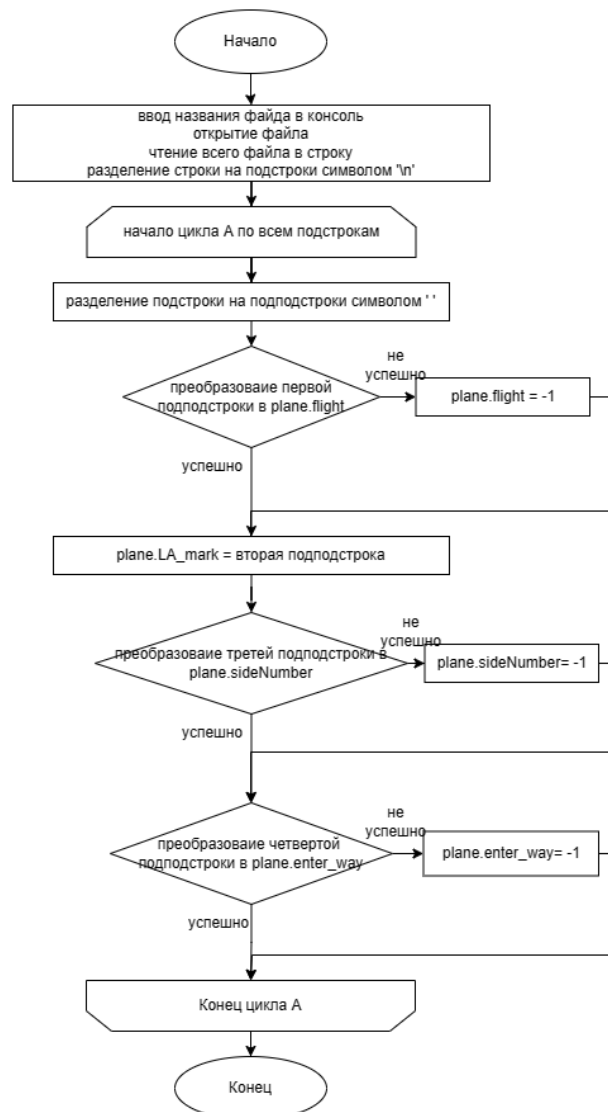


Рис. 2 - Блок-схема функции "readPlanesFromFile"

Функция «ErrorHandler»

1. Назначение

Преобразует самолёт в массив строк

2. Прототип функции

```
char errorHandler(Plane*& planes, int& real_size, String** planesStr, int size, int* cell_size)
```

3. Обращение

```
ret = errorHandler(planeVector, planeVectorSize, string_table, size, cells_size)
```

4. Описание параметров

Идентификатор	Тип	Назначение	Входной\выходной
planeVector	Plane&	Ссылка на массив самолёт	Входной
planeVectorSize	int&	Ссылка на размер самолётов	Входной
String_table	String**	Массив массивов строк	Входной
size	int	Размер пролого массива	Входной
cells_size	int*	Размер всех массивов внутри массива	Входной
result	char	Результат функции	Выходной

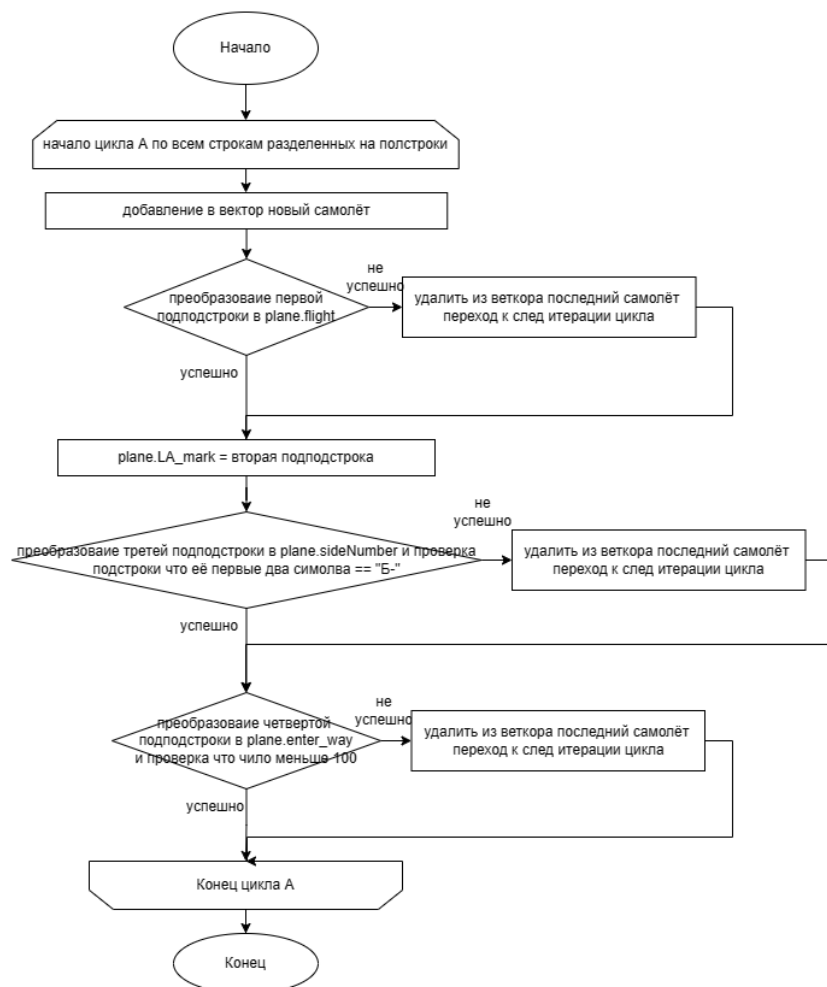


Рис. 3 - Блок-схема функции "ErrorHandler"

Функция «BubbleSort»

1. Назначение

Сортирует индексный массив по высотным эшелонам методом пузырька

2. Прототип функции

```
void BubbleSort(indexItem *sortArray, int
amountCorrectFlights);
```

3. Обращение

```
BubbleSort(sortArray, amountCorrectFlights);
```

4. Описание параметров

Идентификатор	Тип	Назначение	Входной\выходной
sortArray	indexItem[]	Индексный массив	Выходной
size	int	Размер массива	Входной

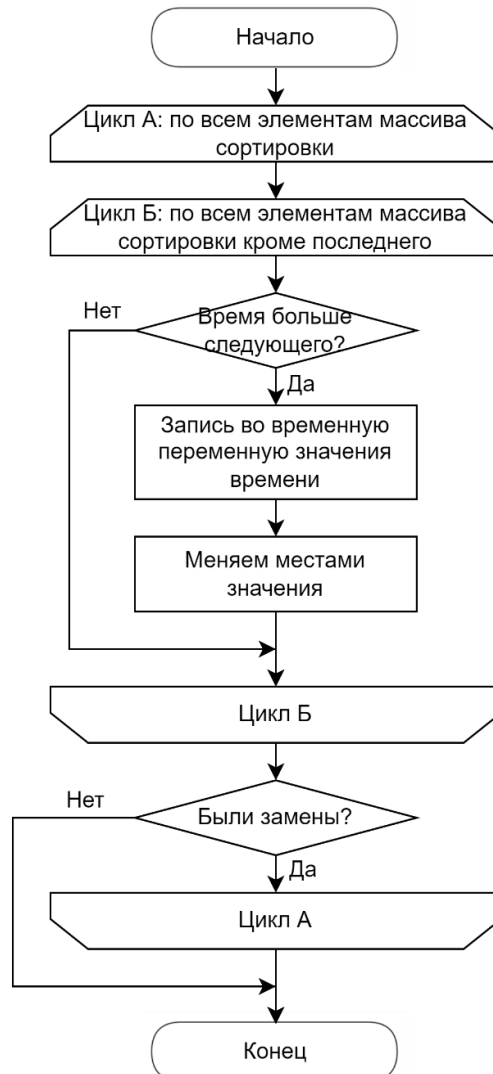


Рис. 4 - Блок-схема функции "BubbleSort"

Код программы

Laba3.cpp

```

/*****
*                                     *
*      КАФЕДРА № 304 1 КУРС      *
*-----*
* Project Type   : Win32 Console Application *
* Project Name   : Laba3 *
* File Name      : Laba3.cpp *
* Language       : C/C++ *
* Programmer(s)  : Закусиллов Л.З., *
* Modified By    : *
* Lit source     : *
* Created        : 29/03/23 *
* Last Revision  : 01/06/23 *
* Comment(s)     : сортировка собственных структур *
*****/
#include "Laba3.h"
#include "String.h"
#include "Table.h"
#include <stdio.h>
#include <fcntl.h>
#include <io.h>

//структура самолёта
struct Plane
{
    int flight = 0; //номер рейса
    String LA_mark = ""; //марка ЛА
    int side_number = 0; //бортовой номер
    int enter_way = 0; //пункт прибытия
};
//конец структуры

String* PlaneToString(Plane& plane)
{
    String* result = new String[4];
    if (plane.flight == -1) {
        result[0] = "ERR";
    }
    else {
        String::IntegerToString(result[0], plane.flight);
    }
    result[1] = plane.LA_mark;
    if (plane.side_number == -1) {
        result[2] = "ERR";
    }
    else {
        String::IntegerToString(result[2], plane.side_number);
        result[2].add(0, 1041);
        result[2].add(1, '-');
    }
    if (plane.enter_way == -1) {
        result[3] = "ERR";
    }
    else {
        String::IntegerToString(result[3], plane.enter_way);
    }
    return result;
}

```



```

//Функция чтения данных из файла
char readPlanesFromFile(Plane*& planes, int& size)
{
    int ret = 0;
    String file_name = "laba3.txt";

    //clear planes
    if (planes != nullptr) {
        delete[] planes;
    }

    //input file name
    printf("    write file name: ");
    ret = writeText(file_name);
    if (ret) {
        return 1;
    }

    //read data from file and echo print
    String buffer;
    ret = readFromFile(buffer, file_name.c_str());
    if (ret) {
        return 1;
    }

    //prepare data
    String* lines = nullptr;
    ret = buffer.split(lines, size, '\n');
    planes = new Plane[size];
    String* line = nullptr;
    int number_of_components = 0;

    //calculate every line
    for (int i = 0; i < size; i++)
    {
        //split line
        String str(lines[i]);
        ret = str.split(line, number_of_components, ' ');

        //check for number of componetnts
        if (number_of_components < 4)
        {
            printf("too few arguments in line: '%d'\n", i + 1);
        }
        if (number_of_components > 4)
        {
            printf("too many arguments in line: '%d'\n", i + 1);
        }

        //check flight
        int temp;
        ret = String(line[0]).toInteger(temp);
        planes[i].flight = temp;
        if (ret) {
            planes[i].flight = -1;
            printf("'flight' corrupted on line '%d'\n", i + 1);
        }

        //check LA_mark
        planes[i].LA_mark = line[1];

        //check side_number
        String sideNumber = line[2];
        if (line[2][0] != 1041 || line[2][1] != '-')
        {
            planes[i].side_number = -1;
        }
    }
}

```

```

        printf("'side number' corrupted on line '%d'\n", i + 1);
    }
    else {
        sideNumber.remove(0);
        sideNumber.remove(0);
        ret = sideNumber.toInteger(temp);
        planes[i].side_number = temp;
        if (ret) {
            planes[i].side_number = -1;
            printf("'side number' corrupted on line '%d'\n", i + 1);
        }
    }

    //check enter_way
    ret = String(line[3]).toInteger(temp);
    planes[i].enter_way = temp;
    if (ret) {
        planes[i].enter_way = -1;
        printf("'enter way' corrupted on line '%d'\n", i + 1);
    }
}
return 0;
}
//Конец функции

```

```

//сортировка пузырьком
char bubbleSort(Plane* planes, unsigned int start, unsigned int end)
{
    char flag = 1;
    for (unsigned int i = start; i < end - 1; i++)
    {
        flag = 1;
        for (unsigned int j = start; j < end - i - 1; j++)
        {
            if (planes[j].flight > planes[j + 1].flight)
            {
                Plane temp = planes[j + 1];
                planes[j + 1] = planes[j];
                planes[j] = temp;

                flag = 0;
            }
        }
        if (flag)
        {
            break;
        }
    }
    return 0;
}
//Конец сортировки пузырьком

```

```

//начало алгоритма
int laba3()
{
    //инициализация переменных
    char ret = 0;
    int size = 0;
    колисечтво структур в файле
    Plane* planes = nullptr;

    //переменная для опознования ошибок
    //переменная хранящие
    //динамичесуй массив структур

```

```

String** t;                                     //массив массивов
строки для таблицы

//конец инициализации переменных

//чтение данных из файла
ret = readPlanesFromFile(planes, size);
if (ret) {
    return -1;
}

t = new String*[size + 1];
t[0] = new String[4];
t[0][0] = L"Номер рейса";
t[0][1] = L"Марка ЛА";
t[0][2] = L"Бортовой номер";
t[0][3] = L"Пункт прибытия";
//эхо печать
for (int i = 0; i < size; i++)
{
    t[i + 1] = PlaneToString(planes[i]);
}
//конец эхо печати
Table a(t, 4, size + 1);
a.drawTable();

//вызов функции сортировки
ret = insertionSort(planes, 0, size);
if (ret) {
    return -1;
}

//печать отсортированных структур
for (int i = 0; i < size; i++)
{
    t[i + 1] = PlaneToString(planes[i]);
}
a.drawTable();
//конец печати

//очистка мусора
if (planes != nullptr)
{
    delete[] planes;
}
//конец очистки мусора

return 0;
} //конец программы

/***** End of main.cpp file *****/

```

Тестирование программы

1. Тестирование некорректных режимов

1. Проверить работу программы, если файла с исходными данными не существует

Ожидаемый результат:

Продолжение работы программы

Результат:

```
write command: 8
  write file name: laba.txt
```


```
write command: 5
```

2. Проверить работу программы, если в файле даны некорректные данные

Ожидаемый результат:

Вывод сообщения об ошибке

Исходные данные:

 laba3.txt – Блокнот

Файл Правка Формат Вид Справка

```
473 ТУ-154М Б-3726 3
366 СУ-113С Б-1236 33
412 СУ-34Д Б-5621 2
366 ТУ-123С В-3276 11
```

Результат:

 Console


```
write command: make laba3
  write file name: laba3.txt
'side number' corrupted on line '4'
```

Номер рейса	Марка ЛА	Бортовой номер	Пункт прибытия
473	ТУ-154М	Б-3726	3
366	СУ-113С	Б-1236	33
412	СУ-34Д	Б-5621	2
366	ТУ-123С	ERR	11

Номер рейса	Марка ЛА	Бортовой номер	Пункт прибытия
366	СУ-113С	Б-1236	33
366	ТУ-123С	ERR	11
412	СУ-34Д	Б-5621	2
473	ТУ-154М	Б-3726	3

2. Тестирование корректных режимов

1. Исходные данные:

 laba3.txt – Блокнот

Файл Правка Формат Вид Справка

```
473 ТУ-154М Б-3726 3
366 СУ-113С Б-1236 33
412 СУ-34Д Б-5621 2
366 ТУ-123С Б-3276 11
```


write command: 8
write file name: laba3.txt

Результат:

Номер рейса	Марка ЛА	Бортовой номер	Пункт прибытия
473	ТУ-154М	Б-3726	3
366	СУ-113С	Б-1236	33
412	СУ-34Д	Б-5621	2
366	ТУ-123С	Б-3276	11

Номер рейса	Марка ЛА	Бортовой номер	Пункт прибытия
366	СУ-113С	Б-1236	33
366	ТУ-123С	Б-3276	11
412	СУ-34Д	Б-5621	2
473	ТУ-154М	Б-3726	3

2. Исходные данные

 laba3.txt – Блокнот

Файл Правка Формат Вид Справка

```
473 ТУ-154М Б-3726 3
366 СУ-113С Б-1236 33
412 СУ-34Д Б-5621 2
366 ТУ-123С Б-3276 121
333 СУ-31Д Б-5621 22
111 ТУ-13С Б-3276 13
```

write command: 8
write file name: laba3.txt

Результат:

Номер рейса	Марка ЛА	Бортовой номер	Пункт прибытия
473	ТУ-154М	Б-3726	3
366	СУ-113С	Б-1236	33
412	СУ-34Д	Б-5621	2
366	ТУ-123С	Б-3276	121
333	СУ-31Д	Б-5621	22
111	ТУ-13С	Б-3276	13

Номер рейса	Марка ЛА	Бортовой номер	Пункт прибытия
111	ТУ-13С	Б-3276	13
333	СУ-31Д	Б-5621	22
366	СУ-113С	Б-1236	33
366	ТУ-123С	Б-3276	121
412	СУ-34Д	Б-5621	2
473	ТУ-154М	Б-3726	3

Вывод

Написал программу, сортирующую записи с использованием индексной сортировки методом пузырька, а также обеспечил входной контроль номера рейса и высоты летательного аппарата.

Чтение данных из файла произвел с использованием функций ввода/вывода языка C++.

Алгоритм параметризован; обмен данными с подпрограммой осуществил через параметры; каждый из наборов исходных данных хранится в отдельном файле.