Здесь будет титульник, листай ниже

СОДЕРЖАНИЕ

1 ПОСТАНОВКА ЗАДАЧИ

Разработать систему, которая демонстрирует возможность использования дружественной функции.

Спроектировать объект, с свойствами в закрытом доступе:

- целого типа, для хранения размерности массива;
- указатель на объект целого типа;
- строкового типа, для хранения наименования объекта.

С параметризированным конструктором. У конструктора есть параметр целого типа. Параметр передает (содержит) значение размерности целочисленного массива. В конструкторе создается целочисленный массив заданной размерности. Вводится и выводиться значение наименования объекта. Вводится и выводится значения элементов.

Объект имеет метод, который возвращает сумму элементов целочисленного массива.

В деструкторе, первоначально выводится значение наименования объекта, а далее значения элементов целочисленного массива и освобождается память, выделенная для массива.

Спроектировать функцию, которая значения элементов массива одного объекта присвоит к элементам массива другого объекта.

Алгоритм конструирования и отработки системы:

1. Объявляется целочисленная переменная, для хранения значения количества объектов.
2. Объявляется целочисленная переменная, для хранения значения размерности массива.
3. Объявляется строковая переменная, для хранения наименования объекта.
4. Могут быть другие объявления.

5. Вводится значение количества объектов.
6. Вводится значение размерности массива.
7. В цикле создаются объекты, согласно введенному количеству.
8. Определяется значение суммы элементов для каждого объекта. Фиксируется объект, с первой минимальной суммой. Этот объект принимается за эталон.
9. В цикле, посредством последовательного вызова дружественной функции значения элементов массива эталонного объекта присваиваются элементам всех остальных объектов.
10. После завершения цикла, созданные объекты удаляются (уничтожаются).

1.1 Описание входных данных

Первая строка:

`«целое число, количество объектов»`

Вторая строка:

`«целое число, размерность массива»`

Начиная с третей строки, имя очередного объекта и значения элементов массивов, согласно количеству объектов:

`«строка» «целое число» «целое число» . . . «целое число»`

Количество целых чисел в этих строках больше или равно количеству размерности массива.

Пример ввода.

```
5
5
obj_2 2 2 2 2 2 2 2
```

```
obj_3 3 3 3 3 3 3 3 3 3
obj_1 1 1 1 1 1
obj_4 4 4 4 4 4 4 4 4
obj_5 5 5 5 5 5
```

1.2 Описание выходных данных

С первой строки, построчно, для каждого объекта:

«строка» «целое число» «целое число» . . . «целое число»

Имя объекта и значения элементов массива, согласно последовательности создания объектов.

Далее, построчно, для каждого объекта:

«имя объекта»: «целое число» «целое число» . . . «целое число»

Имя объекта и значения элементов массива, согласно последовательности создания объектов.

Пример вывода.

```
obj_2  2  2  2  2  2
obj_3  3  3  3  3  3
obj_1  1  1  1  1  1
obj_4  4  4  4  4  4
obj_5  5  5  5  5  5
obj_2: 1 1 1 1 1
obj_3: 1 1 1 1 1
obj_1: 1 1 1 1 1
obj_4: 1 1 1 1 1
obj_5: 1 1 1 1 1
```

2 МЕТОД РЕШЕНИЯ

Для решения задачи используется:

- объект standart класса Fryend* предназначен для хранения указателя на объект, который будет являться стандартом для всех остальных;
- объект one_friend класса Fryend* предназначен для хранения указателя на один из объектов массива;
- функция merge для присваивания значений элементов одного объекта к значениям другого;
- библиотека iostream;
- библиотека vector;
- библиотека limits;
- объекты стандартного потока ввода/вывода данных cin/cout;
- оператор присваивания;
- условный оператор >;
- условная конструкция ветвления if...else;
- оператор цикла со счетчиком for;
- оператор указателя *;
- оператор получения адреса/ссылки &;
- операторы выделения и освобождения динамической памяти new/delete;
- оператор .;
- оператор ->;
- оператор инкремента;
- арифметический оператор суммы.

Класс Fryend:

- свойства/поля:
    - поле массив введенных чисел:

        - наименование — elems;
        - тип — *int;
        - модификатор доступа — private;
    - поле размерность массива:
        - наименование — dim_of_elems;
        - тип — int;
        - модификатор доступа — private;
- функционал:
    - метод Fryend — параметризированный конструктор;
    - метод ~Fryend — измененный деструктор;
    - метод summ — метод, показывающий сумму элементов в поле elems.

3 ОПИСАНИЕ АЛГОРИТМОВ

Согласно этапам разработки, после определения необходимого инструментария в разделе «Метод», составляются подробные описания алгоритмов для методов классов и функций.

3.1 Алгоритм конструктора класса Fryend

Функционал: параметризированный конструктор.

Параметры: целый, dim, размерность массива.

Алгоритм конструктора представлен в таблице 1.

Таблица 1 – Алгоритм конструктора класса Fryend

№	Предикат	Действия	№ перехода
1		присвоение поля dim_of_elems значение параметра dim	2
2		ввод значения поля name	3
3		вывод значения поля name	4
4		выделение памяти для поля elems под dim символов целого типа	5
5		инициализация переменной i со значением 0	6
6	i < dim?	ввод знаения для i элемента массива elems	7
			8
7		вызов оператора инкремента для i	6
8		игнорирование будущих введенных до перехода на новую строку	9
9		инициализация переменной i со значением 0	10
10	i < dim?	вывод " ", i элемент массива elems	11
		переход на новую строку	∅
11		вызов оператор инкремента для i	10

3.2 Алгоритм деструктора класса Fryend

Функционал: измененный деструктор.

Параметры: нет.

Алгоритм деструктора представлен в таблице 2.

Таблица 2 – Алгоритм деструктора класса Fryend

№	Предикат	Действия	№ перехода
1		вывод name, ":"	2
2		инициализация переменной i со значением 0	3
3	i < dim_of_elems?	вывод " ", i элемент массива elems	4
		переход на новую строку	5
4		вызов оператора инкремента для i	3
5		освобождение выделенной под массив elems памяти	∅

3.3 Алгоритм метода summ класса Fryend

Функционал: метод, показывающий сумму элементов в поле elems.

Параметры: нет.

Возвращаемое значение: целое, сумма элементов поля elems.

Алгоритм метода представлен в таблице 3.

Таблица 3 – Алгоритм метода summ класса Fryend

№	Предикат	Действия	№ перехода
1		инициализация переменной sum со значением 0	2
2		инициализация переменной i со значением 0	3
3	i < dim_of_elems?	добавление значения i элемента массива elems к значению переменной sum	4
		возвращение значения переменной sum	∅

№	Предикат	Действия	№ перехода
4		вызов оператора инкремента для i	3

3.4 Алгоритм функции method

Функционал: присваивает значения элементов одного объекта к значениям другого.

Параметры: Fryend&, obj1, объект, чьи значения присваивают, Fryend&, obj2, объект, которому присваивают значения.

Возвращаемое значение: нет.

Алгоритм функции представлен в таблице 4.

Таблица 4 – Алгоритм функции method

№	Предикат	Действия	№ перехода
1		инициализация переменной i со значением 0	2
2	i < dim_of_elems?	присваивание значения i элемента массива elems объекта obj2 значение i элемента массива elems obj1	3
			∅
3		вызов оператора инкремента для i	2

3.5 Алгоритм функции main

Функционал: главная функция программы.

Параметры: нет.

Возвращаемое значение: целое число, идентификатор работоспособности программы.

Алгоритм функции представлен в таблице 5.

Таблица 5 – Алгоритм функции main

№	Предикат	Действия	№ перехода
1		объявление целочисленной переменной amm_of_objects	2
2		объявление целочиселенной переменной dim	3
3		объявление строковой переменной name	4
4		объявление указателя на объект standart класса Fryend	5
5		ввод значений для переменных amm_of_objects, dim	6
6		создание последовательного контейнера vector с объектами типа Fryend* и именем mass_of_friends	7
7		инициализация целочисленной переменной i со значением 0	8
8	i < amm_of_objects?	динамическое выделение памяти под объект one_friend класса Fryend, инициализированного с помощью параметризированного конструктора с параметром dim	9
			11
9		добавление указателя на объект one_friend в mass_of_friends с помощью вызова метода push_back	10
10		вызов оператора инкремента для i	8
11		инициализация целочисленной переменной standasrt со значением 0	12
12	прохождение it по коллекции mass_of_friend		13
			15
13	standart != 0? it > standart?		12
		объекту standart присваивается объект it	14

№	Предикат	Действия	№ перехода
14		значению standart присваивается значение, полученное при вызове метода sum() для объекта it	12
15		инициализация целочисленные переменной i со значением 0	16
16	i < amm_of_objects?	вызов функции merge с переданными параметрами *standart, i объект mass_of_friends	17
			18
17		вызов оператора инкремента для i	16
18		инициализация целочисленной переменной i со значением 0	19
19	i < amm_of_objects?	освобождение выделенной под i элемент mass_of_friends памяти	20
			∅
20		вызов оператора инкремента для i	19

4 БЛОК-СХЕМЫ АЛГОРИТМОВ

Представим описание алгоритмов в графическом виде на рисунках 1-8.

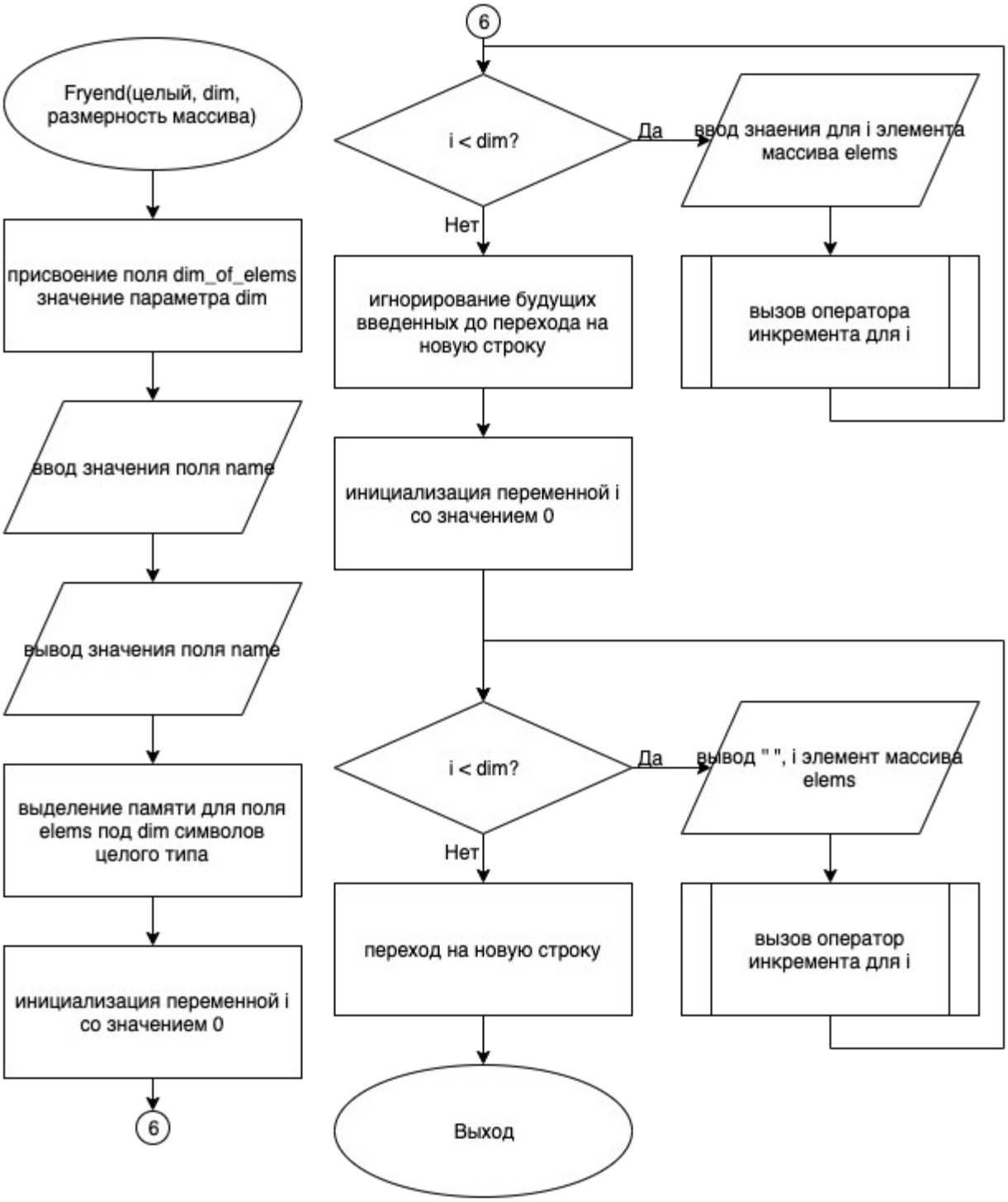

Рисунок 1 – Блок-схема алгоритма

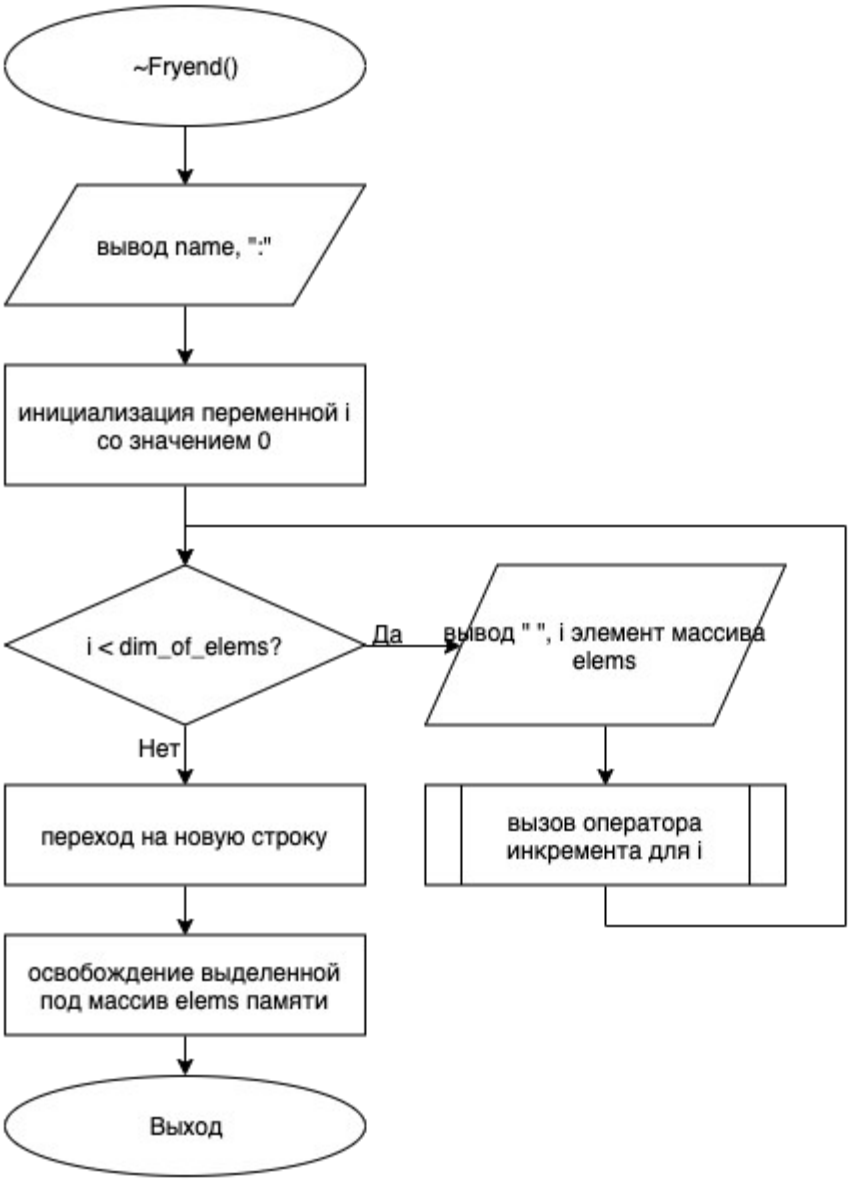

Рисунок 2 – Блок-схема алгоритма

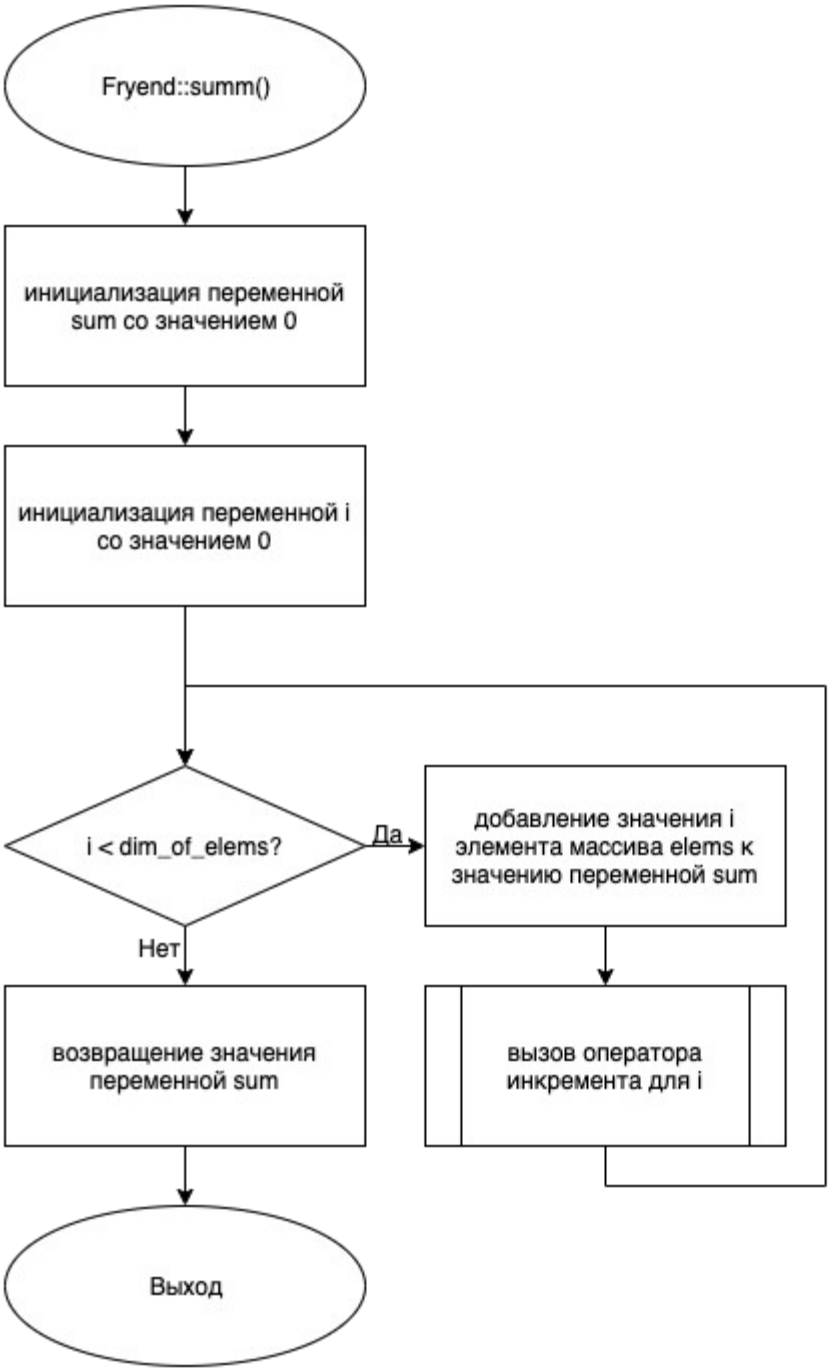

Рисунок 3 – Блок-схема алгоритма

Рисунок 4 – Блок-схема алгоритма

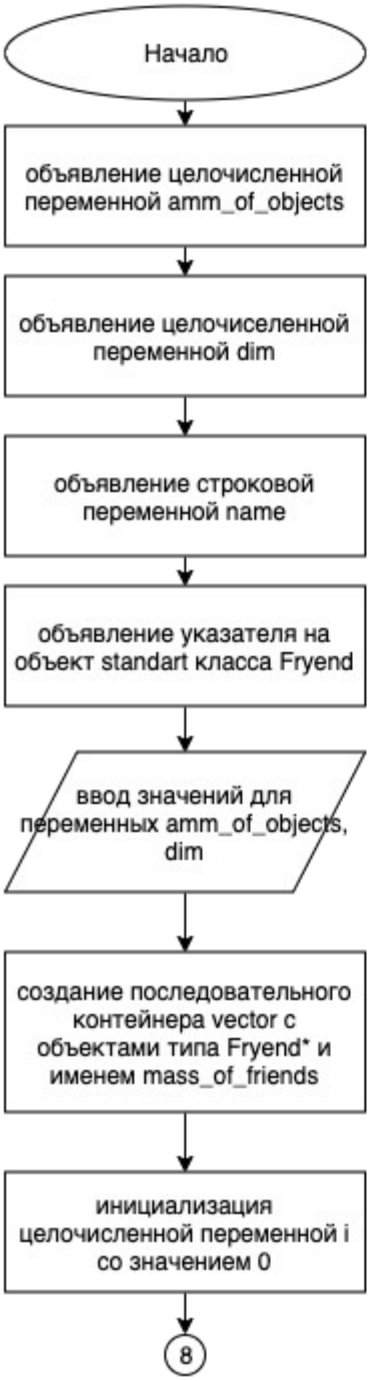

Рисунок 5 – Блок-схема алгоритма

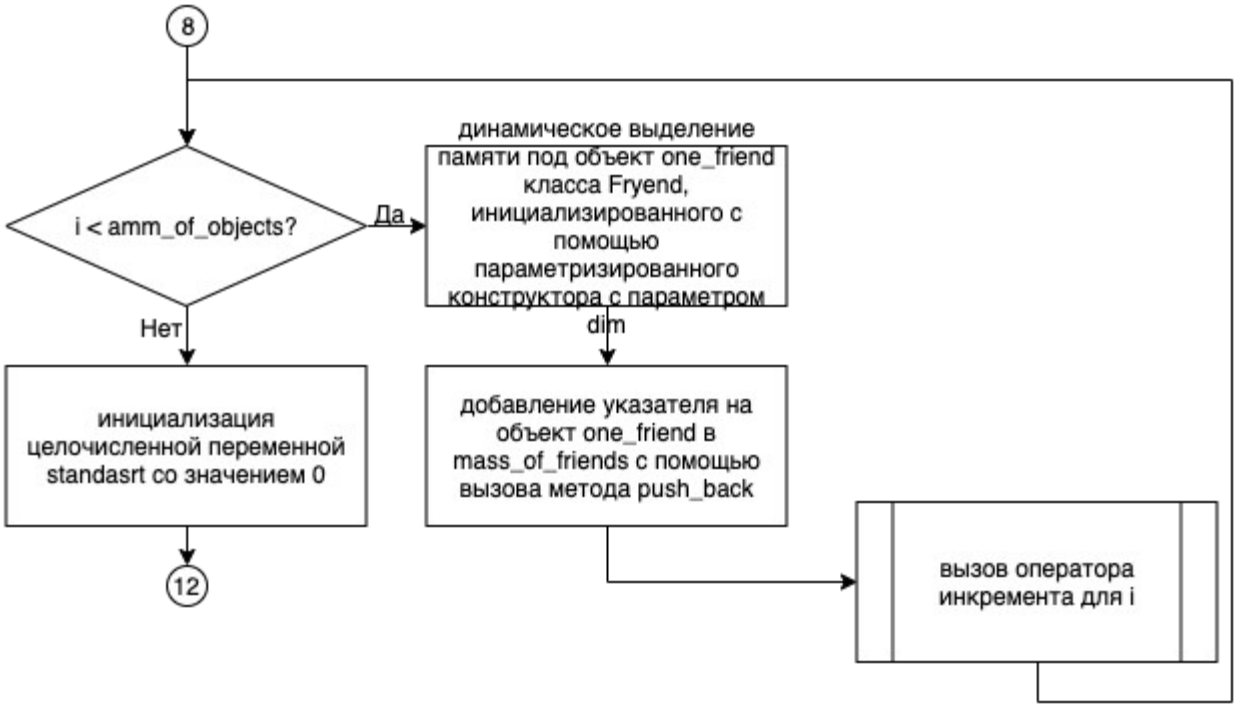

Рисунок 6 – Блок-схема алгоритма

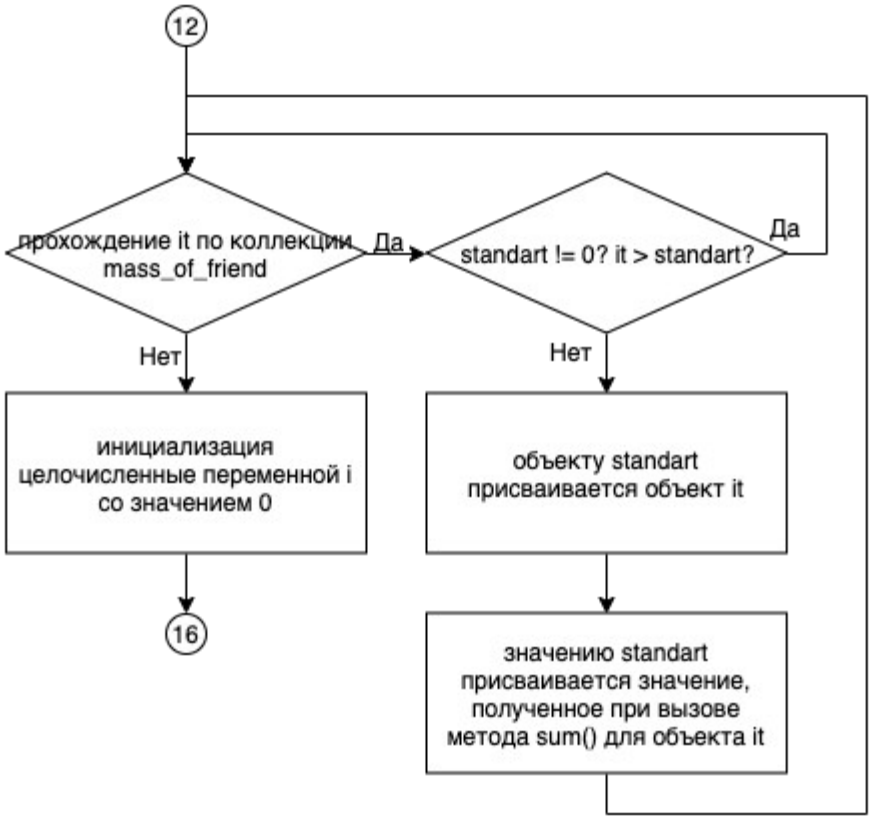

Рисунок 7 – Блок-схема алгоритма

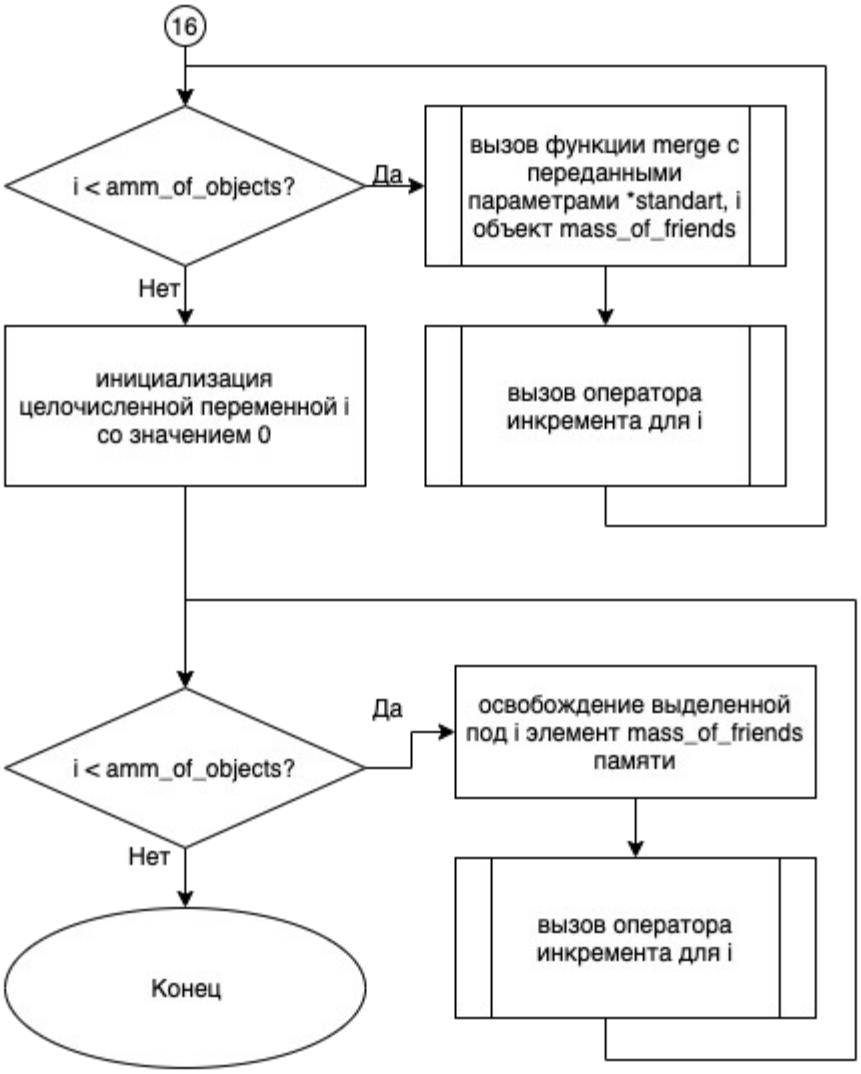

Рисунок 8 – Блок-схема алгоритма

5 КОД ПРОГРАММЫ

Программная реализация алгоритмов для решения задачи представлена ниже.

5.1 Файл Fryend.cpp

Листинг 1 – Fryend.cpp

```
#include "Fryend.h"

Fryend::Fryend(int dim){
   dim_of_elems = dim;
   cin >> name;
   cout << name;
   elems = new int[dim];

   for (int i = 0; i < dim; i++){
      cin >> elems[i];
   }

   cin.ignore(numeric_limits<streamsize>::max(), '\n');

   for (int i = 0; i < dim; i++){
      cout << "  " << elems[i];
   }
   cout << endl;
}

int Fryend::summ(){
   int sum = 0;
   for (int i = 0; i < dim_of_elems; i++){
      sum += elems[i];
   }
   return sum;
}

Fryend::~Fryend(){
   cout << name << ":";
   for (int i = 0; i < dim_of_elems; i++){
      cout << " " << elems[i];
   }
   cout << endl;
   delete elems;
}
```

5.2 Файл Fryend.h

Листинг 2 – Fryend.h

```
#ifndef __FRYEND__H
#define __FRYEND__H

#include <memory>
#include <iostream>
#include <limits>
#include <string>

using namespace std;

class Fryend{
   friend void merge(Fryend& obj1, Fryend& obj2);
   int dim_of_elems;
   int* elems;
   string name;
public:
   Fryend(int dim);
   int summ();
   ~Fryend();
};

#endif
```

5.3 Файл main.cpp

Листинг 3 – main.cpp

```
#include <vector>
#include <string>
#include "Fryend.h"

using namespace std;

void merge(Fryend& obj1, Fryend& obj2){
   for (int i = 0; i < obj1.dim_of_elems; i++){
      obj2.elems[i] = obj1.elems[i];
   }
}

int main()
{
   int amm_of_objects;
   int dim;
   string name;
```

```
    Fryend* standart;
    cin >> amm_of_objects >> dim;
    vector<Fryend*> mass_of_friends;
    for (int i = 0; i < amm_of_objects; i++){
       Fryend* one_friend = new Fryend(dim);
       mass_of_friends.push_back(one_friend);
    }

    /*
    int standard = 0;
    for (auto it : mass_of_friends){
       if (standart != 0 && it->summ() > standard){

       } else {
          standart = it;
          standard = it->summ();
       }
    }
    */

    int standard = numeric_limits<int>::max();
    for (auto it : mass_of_friends){
       int current_sum = it->summ();
       if (current_sum < standard){
          standart = it;
          standard = current_sum;
       }
    }

    for (int i = 0; i < amm_of_objects; i++){
       merge(*standart, *mass_of_friends[i]);
    }

    for (int i = 0; i < amm_of_objects; i++){
       delete mass_of_friends[i];
    }

    return(0);
}
```

6 ТЕСТИРОВАНИЕ

Результат тестирования программы представлен в таблице 6.

Таблица 6 – Результат тестирования программы

Входные данные	Ожидаемые выходные данные	Фактические выходные данные
5 5 obj_2 2 2 2 2 2 2 2 obj_3 3 3 3 3 3 3 3 3 3 obj_1 1 1 1 1 1 obj_4 4 4 4 4 4 4 4 4 obj_5 5 5 5 5 5	obj_2 2 2 2 2 2 obj_3 3 3 3 3 3 obj_1 1 1 1 1 1 obj_4 4 4 4 4 4 obj_5 5 5 5 5 5 obj_2: 1 1 1 1 1 obj_3: 1 1 1 1 1 obj_1: 1 1 1 1 1 obj_4: 1 1 1 1 1 obj_5: 1 1 1 1 1	obj_2 2 2 2 2 2 obj_3 3 3 3 3 3 obj_1 1 1 1 1 1 obj_4 4 4 4 4 4 obj_5 5 5 5 5 5 obj_2: 1 1 1 1 1 obj_3: 1 1 1 1 1 obj_1: 1 1 1 1 1 obj_4: 1 1 1 1 1 obj_5: 1 1 1 1 1

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 19 Единая система программной документации.

2. Методическое пособие студента для выполнения практических заданий, контрольных и курсовых работ по дисциплине «Объектно-ориентированное программирование» [Электронный ресурс] – URL: https://mirea.aco-avrora.ru/student/files/methodichescoe_posobie_dlya_laboratornyh_rabot_3.pdf (дата обращения 05.05.2021).

3. Приложение к методическому пособию студента по выполнению заданий в рамках курса «Объектно-ориентированное программирование» [Электронный ресурс]. URL: https://mirea.aco-avrora.ru/student/files/Prilozheniye_k_methodichke.pdf (дата обращения 05.05.2021).

4. Шилдт Г. C++: базовый курс. 3-е изд. Пер. с англ.. — М.: Вильямс, 2019. — 624 с.

5. Видео лекции по курсу «Объектно-ориентированное программирование» [Электронный ресурс]. АСО «Аврора».

6. Антик М.И. Дискретная математика [Электронный ресурс]: Учебное пособие /Антик М.И., Казанцева Л.В. — М.: МИРЭА — Российский технологический университет, 2018 — 1 электрон. опт. диск (CD-ROM).