Министерство науки и образования РФ

Федеральное государственное автономное образовательное

учреждение высшего профессионального образования

«Санкт-Петербургский государственный электротехнический

университет «ЛЭТИ» им. В. И. Ульянова (Ленина)»

(СПбГЭТУ «ЛЭТИ»)

Факультет компьютерных технологий и информатики

Кафедра вычислительной техники

Отчет по лабораторной работе №6

на тему:

“Обработка одномерных массивов”

по дисциплине “Программирование”

Вариант 5.

Выполнил: Быков И. В.

Группа 6383

Принял: Самойленко В. П.

Санкт-Петербург  
2016 г.

**Цель**

Изучение массивов, индексации в массивах, способов передачи массивов в

функции.

**Теоретические сведения.**

Массив – это структура данных, которая содержит множество значений

одного типа (элементы массива), при этом все элементы массива расположены в памяти последовательно и доступ к конкретному элементу осуществляется через его индекс (порядковый номер). Массивы можно описать следующим образом:

Тип\_элементов\_массива имя\_массива[количество элементов];

Типом элементов массива может быть любой тип С++ (включая массивы).

**Передача массивов в функции**

Используемые в программе массивы могут передаваться при вызове функций в качестве фактических параметров. Можно использовать несколько способов передачи массивов в функцию:

void f1(int arr[15], int n);

void f2(int arr[], int n);

void f3(int \*arr, int n);

void f4(int(&arr)[10]);

В функции f1, f2 и f3 аргумент arr является указателем на начало массива,

поэтому теряется информация о размере массива, длина передаваемого массива может быть любой.

В f4 массив передается по ссылке (как и в случае с обычными переменны-

ми, изменения, вносимые в массив в функции, отразятся на переданном массиве). При передаче по ссылке размер массива можно вычислить с помощью оператора sizeof . Однако в этом случае передавать только массивы, размер которых определен в заголовке функции. Если внутри функции массив не меняется, то для передачи массива в функцию следует использовать константный указатель или ссылку:

void f1(const int arr[15]);

void f2(const int arr[]);

void f3(const int \*arr);

void f4(const int(&arr)[10]);

**Задание**

Рассматривая массивы a[n] и b[m] как последовательности цифр десятичной

записи некоторых неотрицательных чисел, получить массив c[k] – аналогичное представление для суммы этих двух чисел.

**Требования к лабораторной работе.**

1. Обязательно наличие следующих функций: функция ввода массива

из файла, вывода массива в файл/на экран и функция, которая решает постав-

ленную задачу;

1. Не использовать глобальных объявлений, кроме объявления констант;

3. В отчет включить описание спецификации реализованных функций.

**Содержательная постановка задачи.**

Даны 2 числовых массива a[n] и b[m]. При этом каждый элемент обоих массивов есть цифра (0<=a[i]<=9, 0<=b[i]<=9). То есть у нас есть 2 «длинных» числа, требуется найти их сумму. Ответом является массив с[k] – представление их суммы. Каждый элемент массива с[k] – также цифра. Для решения задачи сначала потребуется «зеркально» отобразить a[n] и b[m] (первый элемент массива соответствует последней цифре соответствующего числа, второй элемент – предпоследней, и т.д.). Затем мы складываем числа поразрядно так, как будет описано в алгоритме. И выведем полученный массив в обратном порядке.

**Описание алгоритма.**

Распишем алгоритм работы функции void Find\_Sum(const int \*arr1, const int \*arr2, int len1, int len2, ofstream& outfile).

Мы передали в функцию константные указатели на первый и второй массивы, а так же значение их длин.

1. Зададим max\_len = max(len1,len2)+1 – это длина результирующего массива (+1 – на случай 999+1 = 1000 и т.п.)
2. Объявим массивы arr1\_inv [len\_max] и arr2\_inv[len\_max] и заполним их нулями.
3. Теперь заполним массивы arr1\_inv и arr2\_inv значениями элементов массивов arr1 и arr2 соответственно таким образом, чтобы первый элемент заполняемого массива являлся последним элементом исходного:

for(i = 0; i < len1; i++)

arr1\_inv[i] = arr1[len1 - 1 - i];

1. Найдем сумму. Результат записываем в arr1\_inv. Действуем по алгоритму:
2. Суммируем разряды чисел: arr1\_inv[i] += arr2\_inv[i];
3. Если есть разряд для переноса – переносим: arr1\_inv[i+1] += (arr1\_inv[i]/10);
4. Если разряд для переноса есть – он отсекается: arr1\_inv[i] %=10;
5. Уменьшим размер массива, если главный разряд равен нулю:

if (arr1\_inv[len\_max-1] == 0)

--len\_max;

6) Выведем полученный массив в обратном порядке.

**Тестовые данные.**

|  |  |  |  |
| --- | --- | --- | --- |
| № | arr1[len1] , arr2[len2] | | Сумма arr1\_inv[max(len1,len2)+1] |
| 1 | 0 | 0 | 0 |
| 2 | 9 | 1 | 10 |
| 3 | 123456789 | 987654321 | 1111111110 |
| 4 | 100000 | 50 | 100050 |
| 5 | 50 | 100000 | 100050 |
| 6 | 1 **12** 2 4 | 2 1 3 2 1 3 | Ошибка! |
| 7 | 9 3 2 3 | 4 4 **-3** 2 1 | Ошибка! |
| 8 |  |  | Входной файл пуст! |

**Спецификации функций.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название | Описание | Предусловие | Передается параметр | Что возвращает? |
| int Read\_Arr(istream &in, int \*arr); | Чтение массива из файла |  | указатель на начало массива arr  (0<=arr[i]<=9) | len – длина массива arr |
| void Write\_Arr(ostream &out, const int \*arr, bool start, int len); | Вывод исходных массивов на экран и в файл | start(если true – выводит исходный массив, если false - результат), len –длина массива  (len <= MAX\_SIZE) | константный указатель на начало массива arr (0<=arr[i]<=9) | ничего |
| bool Inspection(const int \*arr, int len); | проверка массива на условие 0<=arr[i]<=9 | len – длина массива  (len <= MAX\_SIZE) | константный указатель на начало массива arr  (0<=arr[i]<=9) | true – если условие выполняется, false – если нет |
| void Find\_Sum(const int \*arr1, const int \*arr2, int len1, int len2, ofstream& outfile); | нахождение суммы двух длинных чисел (каждая цифра - элемент массива).  Алгоритм функции представлен отдельно. | len1 – длина массива arr1, len2 – длина массива arr2  (len1 <= MAX\_SIZE)  (len2 <= MAX\_SIZE) | константные указатели на начала массивов arr1, arr2  (0<=arr[i]<=9) | ничего |

**Текст программы.**

#include <iostream> // cin , cout etc

#include <fstream> // ifstream , outstream etc

using namespace std;

const int MAX\_SIZE = 100;

/\* Функция чтения массива из файла

Передаем указатель на начало массива arr\*

Возвращает длину массива arr \*/

int Read\_Arr(istream &in, int \*arr);

/\* Функция вывода массивов

Выводит полученные массивы и результат работы программы на экран и в файл

Передаем константный указатель на массив arr

Предуcловие: start - определяет, выводить исходные массивы или их сумму

len - размер массива arr\*/

void Write\_Arr(ostream &out, const int \*arr, bool start, int len);

/\* Функция проверки массива на условие 0<=arr[i]<=9

Передаем константный указатель на массив arr

Предусловие: len - размер массива arr

Возвращает true, если условие выполнено, в противном случае - false\*/

bool Inspection(const int \*arr, int len);

/\* Функция нахождения суммы двух длинных чисел (каждая цифра - элемент массива)

Находит сумму по алгоритму, представленному в отчете

Передаем константные указатели на массивы arr1 и arr2

Предусловие: len1 и len2 - длины 1-ого и 2-ого массивов соответственоо\*/

void Find\_Sum(const int \*arr1, const int \*arr2, int len1, int len2, ofstream& outfile);

int main()

{

setlocale(LC\_CTYPE , "rus");

ifstream infile1("fscanf1.txt");

ifstream infile2("fscanf2.txt");

ofstream outfile("fprintf.txt");

if (!infile1)

{

cout << "Не удалось открыть первый файл ввода!\n";

return 1;

}

if (!infile2)

{

cout << "Не удалось открыть второй файл ввода!\n";

return 1;

}

if (!outfile)

{

cout << "Не удалось открыть файл вывода!\n";

return 1;

}

int arr1[MAX\_SIZE];

int arr2[MAX\_SIZE];

int len1 = Read\_Arr(infile1, arr1);

int len2 = Read\_Arr(infile2, arr2);

cout << "В первом массиве считано " << len1 << " элементов: " ;

Write\_Arr(cout, arr1, true, len1);

outfile << "В первом массиве считано " << len1 << " элементов: ";

Write\_Arr(outfile, arr1, true, len1);

cout << "В втором массиве считано " << len2 << " элементов: ";

Write\_Arr(cout, arr2, true, len2);

outfile << "В втором массиве считано " << len2 << " элементов: ";

Write\_Arr(outfile, arr2, true, len2);

if(!Inspection(arr1, len1))

{

cout << "Ошибка! В первом массиве один или более элементов не удовлетворяют условию (0<=a[i]<=9)" << endl;

outfile << "Ошибка! В первом массиве один или более элементов не удовлетворяют условию (0<=a[i]<=9)" << endl;

}

else if(!Inspection(arr2, len2))

{

cout << "Ошибка! В втором массиве один или более элементов не удовлетворяют условию (0<=b[i]<=9)" << endl;

outfile << "Ошибка! В втором массиве один или более элементов не удовлетворяют условию (0<=b[i]<=9)" << endl;

}

else

Find\_Sum(arr1, arr2, len1, len2, outfile);

return 0;

}

bool Inspection(const int \*arr, int len)

{

bool flag = true;

for (int i = 0; i < len; ++i)

if ((arr[i] < 0)||(arr[i] > 9))

flag = false;

return flag;

}

int Read\_Arr(istream &in, int \*arr)

{

int len = 0;

for (; (len < MAX\_SIZE) && (in >> arr[len]); ++len);

return len;

}

void Write\_Arr(ostream &out, const int \*arr, bool start, int len)

{

if (start)

{

for (int i = 0; i < len; ++i)

out << arr[i];

out << endl;

}

else

{

out << "Полученная сумма a + b равна: ";

for (int i = len - 1; i >= 0; --i)

out << arr[i];

out << endl;

}

return;

}

void Find\_Sum(const int \*arr1, const int \*arr2, int len1, int len2, ofstream& outfile)

{

int len\_max = max(len1, len2) + 1;

int arr1\_inv[len1]; // объявим первый инвертированный массив

int arr2\_inv[len2]; // объявим второй инвертированный массив

int i =0;

for(i = 0; i < len\_max; i++) // заполняем массив arr1\_inv нулями

arr1\_inv[i] = 0;

for(i = 0; i < len\_max; i++) // заполняем массви arr2\_inv нулями

arr2\_inv[i] = 0;

for(i = 0; i < len1; i++) // в массиве arr1\_inv лежит инвертированная последовательность цифр массива arr1

arr1\_inv[i] = arr1[len1 - 1 - i];

for(i = 0; i < len2; i++)

arr2\_inv[i] = arr2[len2 - 1 - i]; // в массиве arr2\_inv лежит инвертированная последовательность цифр массива arr2

for(i = 0; i < len\_max; i++) // находим сумму

{

arr1\_inv[i] += arr2\_inv[i]; // алгоритм: суммируем последние разряды чисел

arr1\_inv[i+1] += (arr1\_inv[i]/10); // если есть разряд для переноса - переносим

arr1\_inv[i] %=10; // если разряд для переноса есть - он отсекается (т.е. if a[i]=12 then a[i]=2)

}

if (arr1\_inv[len\_max-1] == 0) // ecли наибольший разряд равен нулю, уменьшим дину массива

--len\_max;

Write\_Arr(cout, arr1\_inv, false, len\_max);

Write\_Arr(outfile, arr1\_inv, false, len\_max);

return;

}

**Результат работы программы.**

1. Вводим 0 и 0

Результат:

В первом массиве считано 1 элементов: 0

В втором массиве считано 1 элементов: 0

Полученная сумма a + b равна: 0

1. Вводим 9 и 1

Результат:

В первом массиве считано 1 элементов: 9

В втором массиве считано 1 элементов: 1

Полученная сумма a + b равна: 10

1. Вводим 123456789 и 987654321

Результат:

В первом массиве считано 9 элементов: 123456789

В втором массиве считано 9 элементов: 987654321

Полученная сумма a + b равна: 1111111110

1. Вводим 100000 и 50

Результат:

В первом массиве считано 6 элементов: 100000

В втором массиве считано 2 элементов: 50

Полученная сумма a + b равна: 100050

1. Вводим 50 и 100000

Результат:

В первом массиве считано 2 элементов: 50

В втором массиве считано 6 элементов: 100000

Полученная сумма a + b равна: 100050

1. Вводим 1 2 34 5 6 и 1 2 3 4 5

Результат:

В первом массиве считано 5 элементов: 123456

В втором массиве считано 5 элементов: 12345

Ошибка! В первом массиве один или более элементов не удовлетворяют условию (0<=a[i]<=9)

1. Оставим файлы пустыми.

Результат:

Входной(-ые) файл(-ы) пуст(-ы)!

**Вывод.**

Научился базовому способу объявления массива. Освоил передачу указателя на массив и константного указателя на массив в качестве параметров функции. Также узнал новые способы вывода данных на экран и в файл, познакомился с длинной арифметикой, которая требуется для сложения чисел, не помещающихся в 64-х разрядный тип данных. Данная лабораторная работа была очень полезна для меня.