Лабораторная работа 2.

Версия 0.8

Целью лабораторной работы является знакомство студентов со статическими одномерными массивами, адресной арифметикой и классическими алгоритмами программирования, такими как поиск минимума и максимума, накопление суммы, накопление произведения, вставка и удаление элементов в массиве, сортировка и др.

Студенты должны научиться

1. описывать одномерные статические массивы;
2. вводить и выводить одномерные массивы;
3. обрабатывать одномерные массивы;
4. передавать одномерные массивы в функции;
5. использовать адресную арифметику для обработки одномерных массивов;
6. измерять время выполнения отдельной функции;
7. выполнять профилирование программы.

Общее задание

1. Исходный код лабораторной работы располагается в отдельной ветке lab_02. В ветке lab_02 для каждой задачи создается папка lab_02_X_Y, где вместо X указывается номер варианта, а вместо Y номер задачи (например, если вы решаете первую задачу третьего варианта, то папка будет называться lab_02_3_1).

2. Исходный код должен соответствовать правилам оформления исходного кода.

3. Для каждой задачи создается отдельный проект в *QT Creator*. Для каждого проекта должно быть два варианта сборки: Debug (с отладочной информацией) и Release (без отладочной информации).
   Замечание
   По согласованию с преподавателем, проводящим практические занятия, вы можете использовать другую среду разработки (например, Microsoft Visual Studio Code), но два варианта сборки проекта нужно предусмотреть в любом случае.

4. Для каждой задачи студентом подготавливаются тестовые данные, которые демонстрируют правильность ее работы. Входные данные должны располагаться в файлах in_z.txt, выходные out_z.txt, где z – номер тестового случая. Тестовые данные готовятся и помещаются под версионный контроль еще до того, как появится реализация задачи.

5. При компиляции программы необходимо использовать ключ “-Wvla”.

6. Для реализации любой из задач этой лабораторной работы вам необходимо выделить несколько осмысленных функций (ввод массива, вывод массива, решение задачи, возможно, какие-то вспомогательные функции). Необходимо предусмотреть обработку ошибочных ситуаций.

7. При вводе массива сначала указывается количество его элементов, затем сами элементы. Ввод неверного количества элементов или недостаточного количества самих элементов считается ошибочной ситуацией.

8. При выводе массива выводятся только его элементы (количество элементов массива выводить не нужно).

9. Ситуация, когда решение задачи не может быть получено, считается ошибочной. Например, нужно подсчитать количество четных элементов массива, а таких элементов в массиве нет. В случае возникновения ошибочной ситуации ваша программа должна не только выдать сообщение, но и вернуть соответствующий код возврата из функции main (0 означает успешное выполнение, любое другое число кодирует ошибку).

10. Реализовав очередную задачу и проверив правильность ее работы, оцените полноту подготовленных тестовых данных на основе процента покрытия кода этими данными. Добейтесь 100% покрытия кода тестовыми данными. Если это невозможно, необходимо это обосновать.

2. Индивидуальное задание

```
Номер задания = Номер в журнале % Количество вариантов.
```

Схема распределения вариантов может быть изменена преподавателем, проводящим практические занятия. Прежде чем приступить к работе над вариантом, уточните этот момент у вашего преподавателя.

Напишите программу, которая запрашивает у пользователя элементы целочисленного статического массива и выполняет его обработку. Максимальное количество элементов, которое может ввести пользователь, равно 10.

Задача 1

Найдите

0. сумму четных элементов массива;
1. произведение нечетных элементов массива;
2. среднее арифметическое отрицательных элементов массива;
3. среднее геометрическое положительных элементов массива.

Задача 2

Сформируйте новый массив из элементов исходного массива. При этом в новый массив помещаются (копируются)

0. элементы исходного массива, которые больше среднего арифметического его элементов;
1. элементы исходного массива, которые являются простыми числами;
2. элементы исходного массива, которые начинаются и заканчиваются на одну и ту же цифру;
3. элементы исходного массива, которые являются числами Армстронга.

Задача 3

0. Удалите из исходного массива все элементы, которые являются числами-палиндромами.
1. Вставьте в исходный массив после каждого элемента, кратного трем, очередное число Фибоначчи (первое число Фибоначчи равно 0, второе – 1).
2. Удалите из исходного массива все элементы, которые являются полными квадратами.

3. Вставьте в исходный массив после каждого положительного элемента этот же элемент, записанный наоборот.

Замечание
Удаление элементов из массива выполняется с сохранением порядка исходных элементов.

Задача 4

Упорядочите исходный массив по возрастанию

0. пузырьком;
1. вставками;
2. выбором.

Задача 5

Выполните обработку массива указанным образом

0. Вычислить значение max(x[0] + x[n-1], x[1] + x[n-2], x[2] + x[n-3], ..., x[(n-1)/2] + x[n/2]), где n размер массива.
1. Вычислить значение x[0]*y[0] + x[1]*y[1] + ... + x[k]*y[k], где x[i] – отрицательные элементы массива a из n элементов, взятые в порядке их следования; y[i] – положительные элементы этого массива, взятые в обратном порядке; k = min(p,q), где p – количество положительных элементов массива a, q – количество отрицательных элементов этого массива.
2. Вычислить значение x[0] + x[0]*x[1] + x[0]*x[1]*x[2] + ... + x[0]*x[1]*x[2] ... x[m], где x[i] - элементы массива x из n элементов, m - индекс первого отрицательного элемента этого массива либо число n-1, если такого элемента в массиве нет.
3. Вычислить значение min(x[0]*x[1], x[1]*x[2],x[2]*x[3], ..., x[n-3]*x[n-2], x[n-2]*x[n-1]), где x[i] - элементы массива x из n элементов.
4. Найти количество различных чисел в файле.

При решении пятой задачи в методических целях нельзя использовать выражение вида a[i] и вообще квадратные скобки. Вместо указанного выражения используется выражение *pa, где pa - указатель на i-ый элемент массива (именно на i-ый элемент, а не выражение вида *(pa + i)). Также нельзя передавать как аргумент размер массива в элементах. Вместо этого предлагается использовать пару указателей: на первый элемент массива и на элемент массива, расположенный за последним. Ситуация, когда эти указатели совпадают, означает пустоту обрабатываемого массива.

Задание 6

На основе задачи 5 проведите сравнение производительности разных способов работы с элементами массива

- использование операции индексации a[i];
- формальная замена операции индексации на выражение *(a + i);
- использование указателей для работы с массивом.

Для этого

- реализуйте функцию, которая выполняет указанное в задаче 5 преобразование массива, используя операцию индексации и количество элементов массива (пусть эта функция называется process_1);
- на основе функции process_1 получить функцию process_2, выполнив замену a[i] на *(a + i);

- реализуйте программу для проведения измерений (ее алгоритм приведен ниже);
- проведите замеры времени.

Алгоритм выполнения измерений

```
1. сформировать случайный целочисленный массив a из na элементов
2. sum = 0
3. в цикле от 1 до N
   // скопируем массив a в массив b
   3.1. b = a
   3.2. nb = na
   // засечем время начала интересующего действия
   3.3. start = get_time
   3.4. process_i(b, nb, ...)
   // засечем время окончания интересующего действия
   3.5. end = get_time
   3.6. sum += (end - start)
4. time = sum / N
// обычно из sum исключают минимальное и максимальное времена
// делить в этом случае нужно на (N - 2)
```

Для замера времени мы будем использовать функцию POSIX функцию gettimeofday, которая возвращает число секунд и микросекунд с 1 января 1970. Пример использования этой функции приведен ниже.

```
...
#include <stdio.h>
#include <inttypes.h>
#include <sys/time.h>

...

int main(void)
{
    ...
    struct timeval tv_start, tv_stop;
    int64_t elapsed_time;

    ...

    gettimeofday(&tv_start, NULL);

    arr_sort(a, n);

    gettimeofday(&tv_stop, NULL);

    elapsed_time = (tv_stop.tv_sec - tv_start.tv_sec) * 1000000LL +
                                        (tv_stop.tv_usec - tv_start.tv_usec);

    // время в микросекундах
    printf("%" PRId64 " μs\n", elapsed_time);

    ...
}
```

Замечание
По согласованию с преподавателем, проводящим практические занятия, для замеров времени может использоваться POSIX функция times.

В отчете приведите таблицу

Количество повторов (N)	Размер массива	a[i]	*(a + i)	Работа с указателями

Проварьируйте количество повторов (десятки, сотни) и размер массива (десятки, сотни, тысячи).

В отчете помимо результатов измерений приведите ответы на следующие вопросы:

1. На что влияет размер массива?
2. Почему приходится выполнять не один замер, а несколько?
3. Какой способ работы с элементами массива оказался самым производительным? Как вы объясняете этот результат?

Задание 7

Прочитайте дополнительные материалы по профилированию (ЛР2_gprof.pdf). Проведите профилирование программы, написанной для решения задачи 4.

Скорее всего, массив из 10 элементов будет обработан очень быстро и результаты профилирования ничего не покажут. Поэтому размер массива нужно увеличить хотя бы до 1000 элементов. Чтобы не вводить такое количество элементов вручную предлагается использовать перенаправление ввода вывода. Для этого нужно создать текстовый файл, в котором на первой строке указанm размер массива, а на следующих строках – элементы этого массива. Создать такой файл можно, например, с помощью программы на Python. После чего запуск программы нужно выполнять следующим образом

```
app.exe < my_data.txt
```

Проанализируйте полученные данные профилирования. Часть этих данных приведите в отчете.

В отчете приведите ответы на следующие задания и вопросы:

1. Совпадают ли ваши представления о времени работы той или иной функции с тем, что вы получили на практике? Если нет, то для какой функции и почему?
2. Увеличьте количество элементов в массиве до 10000, выполните профилирование. Что изменилось? В ответе приведите как сами данные, так и объяснение полученных результатов.
3. Уменьшите количество элементов до 10, выполните профилирование. Что изменилось? В ответе приведите как сами данные, так и объяснение полученных результатов.
4. Изучите, что делают ключи “-O1”, “-O2”, “-O3”. Выполните профилирование вашей программы, увеличив размер массива до 10000 элементов, для каждого из этих ключей. Проанализируйте полученные результаты, сделайте выводы.