Лабораторная работа №8

по дисциплине «Процедурное программирование на C»

Одномерный динамический массив. Адресная арифметика

Кострицкий А. С., Ломовской И. В.

Москва — 2019

Цель работы

1. Научиться работать с одномерными динамическими массивами, использовать функции стандартной библиотеки `malloc`, `calloc`, `realloc`, `free`.

2. Научиться создавать сценарии make для многофайловых проектов.

3. Закрепить навык создания модульных тестов.

Общее задание

Принять с клавиатуры положительное целое n и последовательность n действительных чисел, которые следует записать в динамический массив A.

Вычислить в соответствии с вариантом число $\mu_1 = \mu_1(A)$.

Получить массив A_1, удалив в соответствии с вариантом из динамического массива A набор элементов по признаку. Обратите внимание: даже если на этом шаге несколько раз используется уже вычисленное μ_1, то оно не вычисляется заново. Если массив A_1 сформировать невозможно, или он формируется пустым, считать ситуацию исключительной.

Вычислить в соответствии с вариантом число $\mu_2 = \mu_2(A_1)$.

Получив с клавиатуры номер позиции p, сформировать массив A_2, вставив с сохранением порядка сначала на позицию p массива A_1, а затем добавив в начало и в конец число μ_2. Обратите внимание: даже если на этом шаге используется несколько раз уже вычисленное μ_2, то оно не вычисляется заново. Если на момент вставки на позицию p в массиве меньше, чем $p + 1$ элементов, считать ситуацию исключительной.

Распечатать на экран массив A_2.

Примечания

1. В методических целях при обработке массивов запрещается использовать квадратные скобки. Длину массива разрешается использовать только в функциях работы с памятью.

2. Принять, что под корнем из числа подразумевается арифметический корень из числа.

3. Принять, что под вводом и выводом массива подразумеваются «классические», насколько тут уместно это слово, ввод и вывод элементов массива в строку.

4. Использовать при решении числа двойной точности. Переполнение не контролировать.

5. Для каждой структуры данных в первую очередь реализовать набор подпрограмм группы CDIO — подпрограммы создания, удаления, ввода и вывода.

6. Вынести подпрограммы для каждой структуры данных в отдельный модуль.

7. Следовать правилу Тараса Бульбы для памяти: «Если в подпрограмме есть запрос динамической памяти, то либо в ней же должно осуществляться освобождение памяти, либо в имени подпрограммы должно быть *указание* для программиста на наличие запроса памяти внутри подпрограммы.» В качестве *указаний*, например, можно использовать слова и словосочетания: «allocate», «create», «set length», «new» и другие.

Варианты вычисления μ_1

- (В вариантах $1, 5$) *Среднее геометрическое модулей:*

$$\mu_1(x) = \sqrt[\operatorname{len}(x)]{\left(\prod_{i=0}^{\operatorname{len}(x)-1} |x_i|\right)};$$

- (В вариантах $2, 6$) *Среднее арифметическое:*

$$\mu_1(x) = \sum_{i=0}^{\operatorname{len}(x)-1} \frac{x_i}{\operatorname{len}(x)};$$

- (В вариантах $3, 7$) *Среднее кубическое модулей:*

$$\mu_1(x) = \sqrt[3]{\left(\sum_{i=0}^{\operatorname{len}(x)-1} \frac{|x_i|^3}{\operatorname{len}(x)}\right)};$$

- (В вариантах $4, 8$) *Среднее арифметическое взвешенное:*

$$\mu_1(x) = \frac{\sum_{i=0}^{\operatorname{len}(x)-1}(i+1)^2 x_i}{\sum_{i=0}^{\operatorname{len}(x)-1}(i+1)^2}.$$

Варианты удаления

Удалить из массива A все элементы, которые *меньше* (в варианте 1) или *больше* (в варианте 2) заранее вычисленного числа $\mu_1 = \mu_1(A)$.

Удалить из массива A все элементы, которые *меньше по модулю* (в варианте 3) или *больше по модулю* (в варианте 4) заранее вычисленного числа $\mu_1 = \mu_1(A)$.

Удалить из массива A два элемента, имеющих *минимальную разницу* (в варианте 5) или *максимальную разницу* (в варианте 6) с заранее вычисленным числом $\mu_1 = \mu_1(A)$. Если обнаружены одинаковые кандидаты на удаление, удалять в порядке следования.

Удалить из массива A два элемента, имеющих *минимальную разницу* (в варианте 7) или *максимальную разницу* (в варианте 8) между своим модулем и заранее вычисленным числом $\mu_1 = \mu_1(A)$. Если обнаружены одинаковые кандидаты на удаление, удалять в порядке следования.

Варианты вычисления μ_2

- (В вариантах $1, 5$)

$$\mu_2(x) = \frac{\max(x)\min(x)}{1 + |\max(x)| + |\min(x)|};$$

- (В вариантах $2, 6$)

$$\mu_2(x) = \max(x);$$

- (В вариантах $3, 7$)

$$\mu_2(x) = \min(x);$$

- (В вариантах $4, 8$)

$$\mu_2(x) = \sum_{i=0}^{\operatorname{len}(x)-1} \frac{x_i}{\operatorname{len}(x)}.$$

Пример

Вариант №1: «$\mu_1(x) = \sqrt[\operatorname{len}(x)]{\left(\prod_{i=0}^{\operatorname{len}(x)-1} |x_i|\right)}$, $\mu_2(x) = \frac{\max(x)\min(x)}{1+|\max(x)|+|\min(x)|}$. Удалить из массива A все элементы, которые *меньше* заранее вычисленного числа $\mu_1 = \mu_1(A)$.»

Красным выделены удаляемые элементы.

$$\texttt{In}: \quad n = 7,\ A = (3.1,\ 5.,\ 9.2,\ -1.,\ 3.,\ 2.,\ 10.24),\ p = 2;$$

$$\mu_1(A) = \sqrt[\operatorname{len}(A)]{\left(\prod_{i=0}^{\operatorname{len}(A)-1} |A_i|\right)} = \sqrt[7]{\left(\prod_{i=0}^{6} |A_i|\right)} = \sqrt[7]{8761.344} = 3.657837;$$

$$A_1 = (3.1,\ 5.,\ 9.2,\ -1.,\ 3.,\ 2.,\ 10.24) = (5.,\ 9.2,\ 10.24);$$

$$\mu_2(A_1) = \frac{\max(A_1)\min(A_1)}{1+|\max(A_1)|+|\min(A_1)|} = \frac{10.24 \cdot 5.}{1+|10.24|+|5.|} = 3.152709;$$

$$\texttt{Out}: \quad A_2 = (\mu_2,\ 5.,\ 9.2,\ \mu_2,\ 10.24,\ \mu_2) = (3.152709,\ 5.,\ 9.2,\ 3.152709,\ 10.24,\ 3.152709)\,.$$

Взаимодействие с системой тестирования

1. Решение задачи оформляется студентом в виде многофайлового проекта. Для сборки проекта используется программа make, сценарий сборки помещается под версионный контроль. В сценарии должны присутствовать цель `app.exe` для сборки основной программы и цель `test.exe` — для сборки модульных тестов.

2. Исходный код лабораторной работы размещается студентом в ветви `lab_LL`, а решение каждой из задач — в отдельной папке с названием вида `lab_LL_CC_PP`, где LL — номер лабораторной, CC — вариант студента, PP — номер задачи.

   Пример: решения восьми задач седьмого варианта пятой лабораторной размещаются в папках `lab_05_07_01`, `lab_05_07_02`, `lab_05_07_03`, ..., `lab_05_07_08`.

3. Исходный код должен соответствовать оглашённым в начале семестра правилам оформления.

4. Если для решения задачи студентом создаётся отдельный проект в IDE, разрешается поместить под версионный контроль файлы проекта, добавив перед этим необходимые маски в список игнорирования. Старайтесь добавлять маски общего вида. Для каждого проекта должны быть созданы, как минимум, два варианта сборки: `Debug` — с отладочной информацией, и `Release` — без отладочной информации. Крайне рекомендуем использовать IDE Qt Creator.

5. Для каждой программы ещё до реализации студентом заготавливаются и помещаются под версионный контроль функциональные тесты, демонстрирующие её работоспособность. Входные данные следует располагать в файлах вида `in_TT.txt`, выходные — в файлах вида `out_TT.txt`, где TT — номер тестового случая.

   Под версионный контроль также помещается файл вида `FuncTestsDesc.md` с описанием в свободной форме содержимого каждого из тестов. Вёрстка файла на языке Markdown обязательной при этом не является, достаточно обычного текста.

Разрешается помещать под версионный контроль сценарии автоматического прогона функциональных тестов.

Если Вы используете при автоматическом прогоне функциональных тестов сравнение строк, не забудьте проверить используемые кодировки. Помните, что UTF-8 и UTF-8(BOM) — две разные кодировки.

Пример: функциональные тесты для задачи с двенадцатью классами эквивалентности должны размещаться в файлах `in_01.txt`, `in_02.txt`, ..., `in_12.txt`, `out_01.txt`, `out_02.txt`, ..., `out_12.txt`. В файле `FuncTestsDesc.md` при этом может содержаться следующая информация:

```
# Тесты для лабораторной работы №X
## Входные данные
int a, int b, int c
## Выходные данные
int d, int e
- in_01 -- негативный -- вместо числа a вводится символ
- in_02 -- негативный -- вместо числа b вводится символ
- in_03 -- негативный -- недостаточно аргументов вводятся с консоли
- in_04 -- позитивный -- обычный тест
- in_05 -- позитивный -- вводятся три одинаковых числа
...
```

6. Для каждой подпрограммы должны быть подготовлены модульные тесты, которые демонстрируют её работоспособность.

7. Все динамические ресурсы, которые уже были Вами успешно запрошены, должны быть высвобождены к моменту выхода из программы. Для контроля можно использовать, например, программы valgrind или Dr. Memory.

8. Успешность ввода должна контролироваться. При первом неверном вводе программа должна возвращать код ошибки. Обратите внимание, что даже в этом случае все динамические ресурсы, которые уже были Вами успешно запрошены, должны быть высвобождены.

9. Вывод Вашей программы может содержать текстовые сообщения и числа. Тестовая система анализирует только числа в потоке вывода, поэтому они могут быть использованы только для вывода результатов — использовать числа в информационных сообщениях запрещено.

   Пример: сообщение «Input point 1:» будет неверно воспринято тестовой системой, а сообщения «Input point A:» или «Input first point:» — правильно.

10. Если не указано обратное, числа двойной точности следует выводить, округляя до шестого знака после запятой.

11. *Только для ЛР№8.* Порядок входных данных: n, A, p.

12. *Только для ЛР№8.* Порядок выходных данных: A_2.