Лабораторная работа №10

по дисциплине «Процедурное программирование на C»

Обработка записей с динамическими полями

Кострицкий А. С., Ломовской И. В.

Москва — 2019

Цель работы

1. Научиться работать с записями, имеющими внутри себя указатели на динамические массивы.

Варианты задания

1. О каждом школьнике известны фамилия, пол, дата рождения и оценки за предыдущую четверть. Обрабатывая данные о наборе школьников в виде массива записей, **требуется:**

   (a) Принять с клавиатуры данные о наборе школьников. Признаком окончания ввода считается фамилия «nobody». Формат ввода:

   - Каждое поле вводится с новой строки.
   - Фамилия — набор символов латинского алфавита в любом регистре.
   - Пол — строка «male» или «female» в любом регистре.
   - Дата рождения — строка вида YYYY.MM.DD.
   - Оценки за предыдущую четверть — количество оценок и сами оценки в виде целых чисел, разделённых пробелами.

   (b) Удалить из массива всех девушек старше 17 лет. Текущей датой считать первое ноября 2019 года.

   (c) Добавить к оценкам за четверть всех школьников оценку «3».

   (d) Приняв с клавиатуры один из ключей: `SURNAME, SEX, BIRTHDATE, GRADE`; отсортировать массив устойчиво по соответствующему ключу.

   (e) Распечатать информацию о школьниках в текстовый файл `kids.txt`.

2. О каждом товаре известны артикул, название и количество на складе. Обрабатывая данные о наборе товаров в виде массива записей, **требуется:**

   (a) Принять с клавиатуры данные о наборе товаров. Признаком окончания ввода считается артикул «nothing». Формат ввода:

   - Каждое поле вводится с новой строки.
   - Артикул — набор символов латинского алфавита в верхнем регистре.
   - Название — набор символов латинского алфавита в любом регистре.
   - Количество товара на складе — целое неотрицательное число.

   (b) Добавить в артикул каждого товара его название.

   (c) Удалить из массива все товары артикула «LIPSTICK», которых нет на складе.

   (d) Приняв с клавиатуры один из ключей: `ARTICLE, NAME, COUNT`; отсортировать массив устойчиво по соответствующему ключу.

   (e) Распечатать информацию о товарах в текстовый файл `stockpile.txt`.

3. О каждом эксперименте известны дата проведения, фамилия проводившего и набор показаний прибора за день. Обрабатывая данные о наборе экспериментов в виде массива записей, **требуется:**

   (a) Принять с клавиатуры данные о наборе экспериментов. Признаком окончания ввода считается дата проведения «00.00.0000». Формат ввода:

- Каждое поле вводится с новой строки.
- Дата проведения — строка вида DD.MM.YYYY.
- Фамилия проводившего — набор символов латинского алфавита в любом регистре.
- Набор показаний прибора за день — количество показаний в виде целого числа и сами показания в виде действительных чисел, разделённых пробелами.

(b) Удалить из массива экспериментов все эксперименты, проведённые не позже 1 февраля 1993 года.

(c) Усреднить каждый эксперимент, заменив набор показаний одним показанием.

(d) Приняв с клавиатуры один из ключей: `DATE, TECHNICIAN, DATA`; отсортировать массив устойчиво по соответствующему ключу.

(e) Распечатать информацию об экспериментах в текстовый файл `results.txt`.

4. О каждом студенте известны группа, фамилия, дата рождения и оценки за предыдущий семестр. Обрабатывая данные о наборе студентов в виде массива записей, **требуется:**

(a) Принять с клавиатуры данные о наборе студентов. Признаком окончания ввода считается группа «none». Формат ввода:

- Каждое поле вводится с новой строки.
- Группа — набор символов в любом регистре.
- Фамилия — набор символов латинского алфавита в любом регистре.
- Дата рождения — строка вида YYYY.MM.DD.
- Оценки за предыдущий семестр — количество оценок в виде целого числа и сами оценки в виде действительных чисел, разделённых пробелами.

(b) Удалить из массива студентов группы ИУ7-31Б старше 17 лет. Текущей датой считать первое ноября 2019 года.

(c) Удалить из оценок за семестр всех студентов оценки ниже 4.0.

(d) Приняв с клавиатуры один из ключей `GROUP, SURNAME, GRADE`, отсортировать массив устойчиво по соответствующему ключу.

(e) Распечатать информацию о студентах в текстовый файл `students.txt`.

Примечания

1. При выводе в текстовый итоговый файл каждое поле следует выводить в новой строке. У последнего поля последней записи специально проставлять символ новой строки — не надо реализовывать отдельно вывод последней записи.

2. Реализовать сортировку, используя передачу указателя на функцию-компаратор. Подпрограмма сортировки должна быть одна.

3. Если массив сортируется по строковому полю, то сортировать следует в лексикографическом порядке.

4. Если массив сортируется по полю-массиву, сортировать следует по среднему арифметическому.

5. Принять, что строки в программе не могут быть длинее 256 символов, считая терминальный нуль.

6. Принять, что оперативной памяти целевой машины недостаточно для того, чтобы хранить данные в виде массива статических записей — внутри записей хранить не строки, а указатели на динамические строки, не статические массивы, а динамические массивы. Обратите внимание на то, как ведут себя записи с указателями на динамические поля при копировании и создании.

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

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

9. 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.

Пример

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.

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

11. *Только для ЛР№10.* Порядок входных данных такой, в котором входные данные указаны в формате ввода.

12. *Только для ЛР№10.* Вывод на экран не проверяется. Проверяется только текстовый файл, указанный в задании, формат текстового файла указан в примечаниях.