Факультатив Aldec:
ДЗ №2: Указатели, массивы, строки

Версия 1.0, 23 ноября 2014г.

Простые задачи

1. Напишите функцию readInt, которая пробует считать с консоли целое число. Если пользователь вводит целое число, функция должна записывать данное число в ячейку памяти по указанному адресу и возвращать true. Если же пользователь ввел что-либо другое, не являющееся целым числом, либо прекратил ввод, функция должна возвратить значение false, не пытаясь записывать результат. Ниже приведен пример вызова такой функции:

```
int x;
bool res = readInt( & x );
```

Если ввод будет успешен, то введенное число будет записано в переменной x, и результат функции будет истинным. В противном случае, в переменной x должен сохраниться оригинальный “мусор”, а результат должен быть ложным. Подсказка: функция scanf возвращает количество успешно введенных форматных фрагментов (т.е., при успешном вводе числа по формату “%d”, в случае успеха должна вернуть 1).

2. Напишите функцию inplaceRound, принимающей указатель на значение типа double. Функция должна прочитать число, хранящееся по данному адресу, округлить его до ближайшего целого в соответствии с математическими правилами округления, и перезаписать значение на округленное. Например:

```
double x = 1.2, y = 3.8, z = 5.5;
inplaceRound( & x );
inplaceRound( & y );
inplaceRound( & z );
```

В результате выполнения данного фрагмента переменные x, y и z должны содержать значения 1, 4 и 6 соответственно. Протестируйте функцию при помощи произвольной тестовой программы.

3. Напишите функцию getClosestPowersOf2, принимающую 3 аргумента - целое число, а также два указателя на значения целого типа. Функция должна вычислять ближайшие к первому числу степени двойки - меньшую и большую данного числа. Показатели степеней должны записываться по двум указанным адресам. Например,

```
int x = 23, previousPower, nextPower;
```

```
getClosestPowersOf2( x, & previousPower, & nextPower );
```

В результате, переменная previousPower должна получить значение 4 (2^4=16, 16 < 23), а переменная nextPower - значение 5 (2^5=32, 32>23).

4. Напишите функцию accumulate, которая подсчитает сумму элементов в переданном массиве целых чисел. Перед обработкой массива убедитесь в корректности переданного адреса и количества элементов в массиве при помощи диагностического макроса assert. Проверьте корректность работы функции при помощи произвольной тестовой программы в функции main(). Пример вызова функции:

```
int data[] = { 1, 2, 3, 4, 5 };
int result = accumulate( data, 5 );
```

В результате, переменная result должна содержать значение 15..

5. Напишите функцию iota, которая увеличит на единицу все элементы переданного массива целых чисел. Перед обработкой массива убедитесь в корректности переданного адреса и количества элементов в массиве при помощи диагностического макроса assert. Проверьте корректность работы функции при помощи произвольной тестовой программы в функции main(). Пример вызова функции:

```
int data[] = { 1, 2, 3, 4, 5 };
iota( data, 5 );
```

В результате, массив должен содержать значения { 2, 3, 4, 5, 6 }.

6. Напишите функцию count, которая подсчитает количество элементов переданного массива целых чисел, равных некоторому переданному конкретному значению. Перед обработкой массива убедитесь в корректности переданного адреса и количества элементов в массиве при помощи диагностического макроса assert. Проверьте корректность работы функции при помощи произвольной тестовой программы в функции main(). Пример вызова функции:

```
int data[] = { 1, 2, 3, 2, 1 };
int result = count( data, 5, 1 ); // считать сколько раз 1 в массиве
```

В результате, переменная result должна содержать значение 2, поскольку элемент 1 встречается в массиве дважды.

7. Напишите функцию is_sorted, которая проверяет является ли переданный массив целых чисел упорядоченным по возрастанию либо убыванию. Перед обработкой массива убедитесь в корректности переданного адреса и количества элементов в

массиве при помощи диагностического макроса assert. Проверьте корректность работы функции при помощи произвольной тестовой программы в функции main(). Примеры вызова функции:

```
int data1[] = { 1, 2, 3, 4, 5 };
assert( is_sorted( data1, 5 ) );

int data2[] = { 1, 2, 3, 2, 1 };
assert( ! is_sorted( data2, 5 ) );

int data3[] = { 5, 4, 3, 2, 1 };
assert( is_sorted( data3, 5 ) );
```

Первый массив упорядочен по возрастанию, третий по убыванию. Второй массив не является упорядоченным.

8. Напишите функцию rotate, вращающую элементы в переданном массиве целых чисел относительно указанной базовой позиции. В результате вращения, значения, начиная с указанной базовой позиции, окажутся в начале массива, а значения, предшествовавшие базовой позиции, переместятся в конец массива без нарушения порядка. Перед обработкой массива убедитесь в корректности переданного адреса, количества элементов в массиве и указанной позиции вращения при помощи диагностического макроса assert. Проверьте корректность работы функции при помощи произвольной тестовой программы в функции main(). Пример вызова функции:

```
int data[] = { 1, 2, 3, 4, 5 };
rotate( data, 5, 2 ); // вращаем по позиции 2
```

В результате вызова массив должен содержать данные { 3, 4, 5, 1, 2 }.

9. Пользователь вводит произвольный текст в программу, строку за строкой, завершая ввод нажатием комбинации Ctrl+Z. Программа запоминает строку, содержавшую наибольшее количество знаков препинания. По завершению пользовательского ввода, программа выводит найденную строку. Если ни в одной из строк не было знаков препинания, программа сообщает об этом в явном виде. Если существует более одной строки с одинаковым количеством знаков препинания, программа выводит первую из них.

10. Пользователь вводит в программу две произвольные строки. Программа печатает YES в том случае, если эти строки эквивалентны между собой без учета регистра букв (т.е., считаем маленькие буквы одинаковыми с большиыми). В противном случае программа выводит текст NO.

11. Напишите собственную версию функции strcat из стандартной библиотеки и протестируйте ее на достаточно большом количестве примеров.

12. Пользователь вводит в программу N строк, сообщая о количестве N в начале ввода. Программа “склеивает” все введенные строки в одну единую. Итоговый размер результирующей строки должен быть ограничен (например, 200 символов). Если пользователь пытается ввести данные, которые не помещаются в этот предел, ввод должен быть прерван с соответствующим сообщением об ошибке. Если все строки поместились в результирующий буфер, то программа выводит склеенную строку на экран в обратном порядке.

13. Пользователь вводит в программу строку. Программа определяет является ли данная строка палиндромом (можно прочитать одинаково слева-направо и справа-налево). Регистр букв, пробельные и пунктуационные символы не играют роли при сравнении. Например, палиндромами являются строки “Madam, I am Adam” и “Never odd or even”.

14. Имеется набор шаблонов строк-извинений на нескольких разных языках мира, определенный в программе в явном виде. Пользователь вводит свое имя, а также двухбуквенное обозначение языка (например, “en” для английского, “de” для немецкого, “fr” для французского). Программа пытается найти строку, соответствующую интересующему языку, и распечатать извинение перед пользователем на этом языке. Если такой язык не известен программе, выводится сообщение на английском языке. Например, пользователь вводит:

```
John it
```

Программа отвечает следующей фразой на итальянском языке:

```
John, mi scusi.
```

Если язык не известен программе, выводится сообщение на английском языке:

```
John, excuse me.
```

15. Пользователь вводит в программу N фамилий, сообщая о их количестве N в начале выполнения программы непосредственно перед вводом данных. Затем пользователь вводит интересующий его суффикс. Программа печатает все фамилии, которые заканчиваются на данный суффикс. Например, пользователь ввел следующее:

```
5
Ivanov
Petrov
Sidorenko
Fedorchuk
Kozlov
ov
```

Программа, считав 5 введенных фамилий, находит и распечатывает те из них,

которые заканчиваются на введенный суффикс ov, а именно:

```
Ivanov
Petrov
Kozlov
```

Более сложные задачи

16. Напишите функцию, одновременно вычисляющую радиусы вписанной и описанной окружности для указанного треугольника. Такая функция будет принимать 8 аргументов: 0-1, 2-3, 4-5 - координаты вершин треугольника, 6 и 7 - адреса переменных, куда записать радиус вписанной и описанной окружности соответственно. Функция должна возвращать true и записывать результаты по указанным адресам если переданные вершины образуют треугольник. В случае нарушения правила треугольника, функция должна возвращать false, не модифицируя ячейки памяти для результатов. Постарайтесь максимально повторно использовать вспомогательные функции для работы с треугольниками и окружностями, рассмотренные в предыдущих лекциях.

17. Пользователь вводит в программу N целых чисел, указывая число N в начале перед вводом данных. Программа осуществляет поиск повторяющихся чисел в массиве, и замещает их соседними не повторяющимися. Предположим, повторяющиеся числа встретились M раз, M<N. Тогда программа должна переупорядочить массив таким образом, чтобы в позиция от 0 до (N-M-1) включительно оказались все уникальные числа. Эти числа необходимо вывести на экран. Содержимое остальных ячеек в памяти массива не интересует. Например, пользователь вводит число N=15, и следующий набор данных:

```
1   2   3   4   5   2   4   6   7   1   8   9   3   0   12
```

Программа обнаруживает M=4 повторяющихся чисел, переупорядочивает массив и выводит следующий результат:

```
1   2   3   4   5   6   7   8   9   0   12
```

Для ввода и вывода массивов рекомендуется использовать готовые функции scanArray и printArray, рассмотренные в лекции. Алгоритм, удаляющий повторяющиеся числа, рекомендуется оформить в виде отдельной функции, принимающей массив, и возвращающей адрес элемента, следующего за последним неповторяющимся в результате обработки:

```
int * uniquify ( int * pData, int size );
```

Для массива из указанного примера должен возвращаться адрес массива + 11 позиций. Количество неповторяющихся чисел легко определить разницей между полученным указателем и началом массива.

18. Самостоятельно изучите поведение функции strtok из стандартной библиотеки. Реализуйте и протестируйте собственную реализацию такой функции.

19. Пользователь вводит текст, содержащий слова и целые числа, разделяемые пробелами и знаками препинания. Программа обнаруживает числа и переводит их в прописной вид на английском языке. Затем исходный текст печатается на экране, при этом все числа в нем заменены прописными эквивалентами. Например:

    Ввод: An amount of 1200 US dollars was sent to the account.
    Вывод: An amount of one thousand two hundred US dollars was sent to the account.

20. Пользователь вводит в программу информацию о покупах товаров в следующем формате:

    Paper 50.00
    Milk 12.50
    Toothpaste 35.60

    Программа анализирует введенные данные и распечатывает:
    - самую дешевую и самую дорогую покупку
    - общую сумму трат
    - товары, которые покупались более одного раза
    - разные товары, которые имеют одинаковую цену.

Подсказка: простейший способ хранения такой информации - два массива (массив строк с именами и массив чисел с ценами) с согласованными индексами. Т.е., для введенного примера по индексу 1 в первом массиве будет строка “Milk”, а во втором по этому же индексу будет храниться цена 12.50.

Для упрощения решения задачи можно ограничить ввод некоторым фиксированным количеством N записей о купленных товарах. Пользователь может ввести и меньшее количество данных, не обязательно именно равное N, однако программа не должна допускать превышение этого предела.

Отправка результатов

- Решения следует присылать по электронной почте по адресу zaychenko.sergei@gmail.com

- Разрешается прорешать любое число задач на усмотрение студента.

- **Важно**: Перед отправкой файлы решения следует заархивировать, при этом преподавателя интересуют только файлы исходного кода. Никакие бинарные файлы, файлы проектов среды разработки присылать не следует. При соблюдении данных условий размер архивного файла не привысит нескольких килобайт.

- При решении задач следует избегать типовых ошибок и неточностей, чтобы сэкономить собственное время и время преподавателя:

    - не используйте предварительно откомпилированные заголовочные файлы (изначально создавайте проекты только для пустых консольных приложений);

    - не используйте академическую идиому в стиле вызова getch() либо system(“pause”) в конце программы;

    - при использовании потокового ввода вывода, не следует применять глобальных директив using namespace std - старайтесь избегать подобной конструкции вообще, либо, как минимум, делать ее локальной в области непосредственного использования идентификаторов из библиотеки.