Лабораторная работа №6. Строки, файлы, структуры

Особенности работы со строками

Строки используются для хранения текстовой информации. Для обработки больших объемов текстовой информации можно использовать массивы строк.

В C++ есть два вида строк:

- C-строки (их еще называют строки старого стиля).
- Класс стандартной библиотеки C++ string (строки нового стиля).

Для нединамических строк удобно задавать ее длину с помощью именованной константы. Однако нужно учитывать, что один символ на конце строки всегда занимает символ ее конца (нуль-символ). Поэтому, если необходимо задать строку длиной в n символов, то ее длина должна быть равна n + 1. Выход за границы строки и отсутствие нуль-символа являются распространенными причинами ошибок в программах.

Работа с динамическими строками похожа на работу с динамическими массивами. Длина динамической строки может быть переменной. Динамические строки нельзя инициализировать при создании. Динамические строки рекомендуется использовать только при необходимости.

Для строк старого стиля не определена операция присваивания, поскольку строка является не базовым типом данных, а массивом специального вида. Присваивание выполняется при помощи функций стандартной библиотеки или посимвольно «вручную» (что менее предпочтительно, т.к. чревато ошибками).

Кроме привычных функций копирования и нахождения длины строки, библиотека string предоставляет также различные функции для сравнения строк и подстрок, объединения строк, поиска в строке символа и подстроки и выделения из строки лексем. В библиотеке определен целый ряд функций, проверяющих принадлежность символа какому-либо множеству, например, множеству букв, разделителей, знаков пунктуации, цифр и т.д. Эти функции можно узнать, обратившись к справочной литературе (например, Приложение 6 учебника, описание заголовочного файла <cstring> стр. 414-415, алфавитный перечень стр. 416-446).

Для консольного ввода-вывода строк используются либо объекты cin и cout, либо функции библиотеки gets, scanf и puts, printf. Ввод строки с помощью операции >> выполняется до первого пробельного символа. Для ввода строки, содержащей пробелы, можно использовать либо методы getline или get класса istream, либо функции библиотеки gets и scanf. Функциями семейства printf удобнее пользоваться в том случае, если в одном операторе требуется ввести или вывести данные различных типов. Если же работа выполняется только со строками, проще применять специальные функции для ввода-вывода строк gets и puts.

Особенности работы с файлами

В качестве файлов рекомендуется брать тексты на русском или английском языке, сохраненные в формате текстового документа (*.txt). Предварительно файл необходимо подготовить. Во-первых, нужно найти (в Интернете, электронной книге, файлах справки, где-то еще) текст подходящего размера. Обычно хватает текста из нескольких тысяч символов, но для отладки можно использовать небольшие файлы. Во-вторых, необходимо отформатировать строки так, чтобы в одной было не больше 80 символов (это число является неофициальным стандартом размера строки текстового файла).

Ввод-вывод из файла может выполняться с помощью либо объектов классов ifstream и ofstream (ввод/вывод в стиле C++, теоретический материал в Учебнике, Глава 10), либо функций ввода/вывода в стиле C (заголовочный файл <stdio.h>, алфавитный перечень функций в Учеб-

нике на стр. 411-413, подробное описание стр. 416-446). Примеры работы с файлами см. в Практикуме, семинары 5 и 6.

Помните, что посимвольное чтение из файла неэффективно.

Особенности работы с русским языком в Windows

Существует некоторая проблема с использованием русского языка в консольных приложениях. Причины этого объясняются в Практикуме на стр. 17. Для того, чтобы символы кириллицы отображались корректно, нужно использовать один из следующих трех способов:

1. Использование специализированных функций.

Наиболее универсальный способ, работает всегда и не требует дополнительных настроек среды. Необходимо добавить в текст программы следующий код:

```
char bufRus[256];
char* Rus(const char* text) {
      CharToOemA(text, bufRus);
      return bufRus;
}
```

Для того, чтобы можно было использовать функцию `CharToOemA()`, подключите заголовочный файл: `#include<windows.h>`

Функция Rus() получает в качестве аргумента строку, в качестве результата получается строка в другой кодировке, которая корректно отображается на экране.

Если необходимо считывать с клавиатуры строку на русском, то следует использовать функцию `OemToCharA()`. Она работает аналогичным образом и должна получать два аргумента, поэтому для удобства можно самостоятельно сделать функцию, аналогичную Rus().

2. Локализация.

Локализация – это адаптация программного обеспечения к культуре какой-либо страны. В нашем случае при написании программы локализация означает возможность использования национальных языков в пользовательских интерфейсах программ. Нас интересует русский язык. Существует несколько вариантов команд для подключения русской локализации, например, такие:

```
setlocale( LC_CTYPE,""); //корректный вывод
setlocale(LC_ALL, "C"); //корректное чтение с клавиатуры
```

Не забудьте подключить заголовочный файл `#include <clocale>`

Проблема локализации состоит в следующем. Для того чтобы ввод/вывод работал корректно, необходимо писать соответствующие строки каждый раз перед командами ввода или вывода. Если этого не делать, в некоторых случаях локализация отключается.

3. Настройка среды программирования.

Подключите заголовочный файл `#include<windows.h>`, в начало функции main() добавьте следующие строчки:

```
SetConsoleCP(1251);
SetConsoleOutputCP(1251);
```

После этого необходимо запустить программу и в свойствах появившегося окна выбрать шрифт `Lucida Console.` Теперь при вводе/выводе символов кириллицы не будет проблем.

> **Замечание:** Настройки шрифта сохраняются и при следующих запусках консольных приложений на этом компьютере, однако если в программе нет указанных выше строчек, русский язык работать не будет.

Особенности работы со структурами в C++

Чаще всего структуры в C++ применяются для логического объединения связанных между собой данных. В структуру можно объединять данные различных типов.

Элементы структуры называются полями. Поля могут быть любого основного типа, массивом, указателем, объединением или структурой. После описания структурного типа обязательно ставится точка с запятой. Для обращения к полю используется операция выбора: «точка» при обращении через имя структуры и «->» при обращении через указатель.

Структуры одного типа можно присваивать друг другу, однако ввод/вывод структур выполняется поэлементно. Структуры, память под которые выделяет компилятор (нединамические структуры), можно инициализировать перечислением значений их элементов.

В частности, при работе с данными удобно использовать бесконечный цикл с принудительным выходом. Однако возможен вариант с использованием специальной переменной-флага, определяющей условия продолжения цикла. Можно для обеспечения принудительного выхода из цикла использовать нажатие некоторой клавиши, например, Esc, поскольку такой интерфейс наиболее удобен для пользователя.

Сортировка массивов структур выполняется практически аналогично обычной сортировке массивов. Разница состоит в том, что сравниваются между собой только ключевые поля структур (те поля, по которым происходит сортировка), а значениями меняются сами элементы.

Задания для всех

Все задания выполняются с использованием функций, количество таких функций для разных задач может различаться. Если на входные данные и результаты накладываются какие-то ограничения, отразить это в программе.

Номер варианта можно узнать у преподавателя. Работы, выполненные по чужому варианту, не принимаются. Номер варианта остается постоянным для всех лабораторных работ. Все задания выполняются с использованием функций.

1. (2 балла) Имеется текстовый файл, содержащий не менее 100 строк длиной до 80 символов. Составить словарь из слов, содержащихся в этом файле. Если слово входит в текст несколько раз, в словаре оно должно быть записано только однажды. Все слова должны быть в нижнем регистре. Словарь должен быть отсортирован в алфавитном порядке и сохранен в новом текстовом файле. Если словарь построен, но не отсортирован, дубли не удалены, за задание можно получить не более одного балла.
2. (2 балла) Имеется текстовый файл «test1.txt», содержащий некоторое количество строк длиной до 80 символов.

Замечание: если задача будет решена только для одной строки, за задание можно получить не больше одного балла. Готовое задание предоставлять преподавателю вместе с примером файла, которые содержит все возможные варианты исходных данных. В любом случае в файле должно быть не менее 10 строк связного текста (из них не менее одной строки, содержащей ровно 80 символов).

Найти и вывести на экран:
Вариант 1: количество слов в каждой строке.
Вариант 2: строки, содержащие числа.
Вариант 3: все слова, начинающиеся с гласных.
Вариант 4: цитаты, т.е. слова и фразы, заключенные в кавычки.
Вариант 5: весь текст в обратном порядке.
Вариант 6: самое длинное слово каждой строки.
Вариант 7: все слова, которые начинаются и оканчиваются одной и той же буквой.
Вариант 8: сначала все вопросительные, потом все восклицательные предложения.

Подготовлено Латухиной Е.А.

Вариант 9: количество слов, которые содержат хотя бы одну букву «а».
Вариант 10: все слова, начинающиеся и заканчивающиеся на гласные.
Вариант 11: количество слов, которые содержат ровно три буквы «а» (не обязательно подряд!).
Вариант 12: самое короткое слово каждой строки.
Вариант 13: количество однобуквенных слов.
Вариант 14: весь текст, меняя местами каждые два соседних слова.
Вариант 15: все слова длины n (n вводится с клавиатуры).
Вариант 16: все слова длиной меньше чем n (n вводится с клавиатуры).
Вариант 17: все слова, которые начинаются с заглавных букв.
Вариант 18: все слова длиной больше чем n (n вводится с клавиатуры).
Вариант 19. количество гласных букв в каждой строке.
Вариант 20: все строки в обратном порядке.

3. (3 балла) Решите задачу своего варианта.

Необходимо описать структуру с заданным именем, полями и содержащую следующие методы:
Read – ввод с клавиатуры.
Init – проверка корректности значений полей.
Display – вывод на экран данных из структуры в «правильном виде».
Кроме того, необходимо реализовать внешнюю функцию, которая выполняет описанное для конкретного варианта задание.

Вариант	Имя структуры и назначение	1 поле	2 поле	Задание
1	complex, комплексные числа	Re	Im	Вычитание
2	complex, комплексные числа	Re	Im	Модуль
3	complex, комплексные числа	Re	Im	Умножение
4	complex, комплексные числа	Re	Im	Деление
5	triangle, прямоугольный треугольник	a	b	Площадь
6	triangle, прямоугольный треугольник	a	b	Косинус alpha
7	triangle, прямоугольный треугольник	a	b	Гипотенуза
8	triangle, прямоугольный треугольник	a	b	Высота к стороне c
9	triangle, прямоугольный треугольник	a	b	Радиус описанной окружности
10	time, хранение времени	hour	minute	Вычесть минуты
11	time, хранение времени	hour	minute	Разность
12	time, хранение времени	hour	minute	Добавить минуты
13	time, хранение времени	hour	minute	Перевод в минуты
14	money, денежный тип	rub	kop	Перевод в копейки
15	money, денежный тип	rub	kop	Сложение
16	money, денежный тип	rub	kop	Увеличение на процент
17	money, денежный тип	rub	kop	Добавить копейки
18	money, денежный тип	rub	kop	Вычитание
19	linsolve, линейное уравнение	A	B	Найти корень Ax+B=0
20	linsolve, линейное уравнение	A	B	По x найти y