Некоммерческое акционерное общество

«Алматинский Университет Энергетики и Связи имени Гумарбека Даукеева»

Кафедра Автоматизация и управление



**Лабораторная работа №1**

**Работа с символьной информацией**

По дисциплине: Системное Программное Обеспечение и Программирование (СПОИП)

Специальность: Автоматизация и управление

Выполнил(-а): Култаев Н.Е.

Группа: АУ-18-8

Принялa: Сябина Н.В.

« » 2020 г. (оценка) (подпись)

Содержание

[Введение 3](#_Toc51840143)

[1 Лабораторная работа. Работа с символьной информацией 4](#_Toc51840144)

[2. Листинг программы и комментарии к лабораторной работе № 1: 7](#_Toc51840145)

[3. Вывод программы через компилятор gcc 11](#_Toc51840146)

[Заключение 12](#_Toc51840147)

[Список литературы 13](#_Toc51840148)

[Контрольные вопросы 14](#_Toc51840149)

Введение

Величина типа «символ» может принимать значения любых символов компьютерного алфавита. Символьная величина занимает 1 байт памяти, в котором хранится код этого символа, соответствующий используемой кодовой таблице.

Символьная константа записывается между апострофами.

Первые компьютеры умели обрабатывать только числовую информацию. Теперь же большая их часть занята обработкой информации, представленной в виде текста. Чтобы это стало возможно, символы текста (буквы, цифры, знаки препинания и т.д.) стали кодировать при помощи чисел. Для этого создаётся таблица, в которой каждому символу алфавита, при помощи которого в компьютере записываются тексты, присваивается уникальный номер. Получается кодовая таблица. Число символов в этой таблице определяется количеством памяти, выделяемым для кодирования одного символа.

1 Лабораторная работа. Работа с символьной информацией

Цель: получить практические навыки в работе с массивами и указателями языка C, научиться обеспечивать функциональную модульность.

1.1 Постановка задачи

Создать функцию Delete(s,n,l), предназначенную для удаления из строки s подстроки, начиная с позиции n, длиной l. Предусмотреть обработку ошибок в задании параметров и особые случаи. Разработать два варианта функции, используя традиционную обработку массивов и используя адресную арифметику.

1.2 Метод решения

Строка символов в C представляется в памяти как массив символов, в конце которого находится байт с кодом 0 — признак конца строки. Строку, как и любой другой массив, можно обрабатывать либо традиционным методом как массив с использованием операции индексации, либо через указатели с использованием операций адресной арифметики. При работе со строкой как с массивом нужно иметь в виду, что длина строки заранее неизвестна, поэтому циклы целесообразно организовывать не со счетчиком, а до появления признака конца строки.

Функция должна реализовывать поставленную задачу — и ничего более. Это означает, что функцию можно будет, например, перенести без изменений в любую другую программу, если спецификации функции удовлетворяют условиям задачи. Это также означает, что при ошибочном задании параметров или при каких-то особых случаях в их значениях функция не должна аварийно завершать программу или выводить какие-то сообщения на экран, но должна возвращать какое-то прогнозируемое значение, по которому та функция, которая вызвала нашу, может сделать вывод об ошибке или об особом случае.

Определим состав параметров функции

char\* Delete (s, n, l);

где ; s — подстрока, которую надо удалить; n — номер первого символа в строке const\_string, с которого начинается подстрока (нумерация символов ведется с 1); l — длина подстроки.

Возможные возвращаемые значения функции установим: 1 (задание параметров правильное) и 0 (задание неправильное). Эти значения при обращениях к функции можно будет интерпретировать как «истина» или «ложь».

Обозначим через i индекс в строке const\_string. Тогда при задании параметров возможны следующие случаи:

а) i < n+l— полностью правильное задание;

б) ch != const\_string[i++] - Символы не совпадают;

в) const\_string[i] = '\0' и ch != 0 - Слово длиннее строки, либо позиция указана не верно. Неправильное задание

г) i > n+l — неправильное задание, выходная строка будет пустой, сообщение об ошибке.

1.3 Описание логической структуры

Программа состоит из одного программного модуля — файл main.cpp. В состав модуля входят пять функций: main, Delete, DeleteNew, isAllCorrect и Error\_ptr. Глобальные переменные в программе:

i, itter, cc - иттераторы для пробегания по строке/ам;

code\_error - Код ошибки для вывода соответствующего сообщения;

const\_string - Исходная строка;

edited\_string - Измененная строка.

Макроконстантой N определена максимальная длина строки — 256. Листинг программы приводится в приложении А.

Функция main является главной функцией программы, она предназначена для ввода исходных данных, вызова других функций и вывода результатов. В функции определены переменные: L[N] — входная строка; pos — номер символа, с которого должна начинаться выходная строка; lут — длина выходной строки.

Функция запрашивает и вводит значение входной строки, номера символа и длины. Далее она вызывает функцию Delete, передавая введенные значения как параметры. Функция Delete возвращает измененную строку, на экран выводятся выходная строка, если же возвращает нулевой байт — выводится сообщение об ошибк. Далее вызывается функция DeleteNew, если Delete не вернула ошибку, и выводится тоже самое, что и при Delete.

Функция Delete выполняет поставленное задание методом массивов. Ее параметры: s — подстрока, представленный в виде массива неопределенного размера; n и l.

Во-первых, функция проверяет строку на наличие ошибки с помощью предиката isAllCorrect, при возвращении false выполняется функция Error(), которая выводит сообщение об ошибке, если же вернулась true, то все параметры введены верно и можно запускать процедуру удаления подстроки.

Так как используются глобальные переменные, то для начала обнуляется i для пробегания по исходной строке и ch возвращается на начало строки, чтобы зайти в while.

В while, если позиция находится в «удаляемой зоне», то накручиваем i, как бы пропуская этот промежуток. Если же нет, то выполняются две операции:

1. Присваивает ch позицию I;
2. Записывает символы в новую измененную строку

При прохождении цикла возвращается измененная строка.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| №  теста | Входные данные | | | | Выходные данные |
| **const\_string** | **s** | **n** | **l** | **edited\_string** |
| 1 | 012345 | 12 | 2 | 2 | 23 |
| 012345 | 1234 | 2 | 4 | 05 |
| 012345 | 012345 | 1 | 6 | Нулевой байт |
| 2 | 012345 | 45 | 5 | 2 | 0123 |
| 012345 | 12345 | 2 | 5 | 0 |
| 012345 | 23 | 3 | 2 | 0145 |
| 3 | 012345 | 123 | 8 | 2 | Ошибка |
| 4 | 012345 | 1254 | -1 | 2 | Ошибка |
| 012345 | 13546 | 5 | 0 | Ошибка |
| 012345 | 13544 | 5 | -1 | Ошибка |

2. Листинг программы и комментарии к лабораторной работе № 1:

/\*

\*\* What I need to do? \*\*

\*\*\* Создать функцию Delete(s,n,l)

\*\*\* удаление из строки s подстроки, начиная с позиции n, длиной l

\*/

/\*

\*\*Explanes of the variables names\*\*

\*\*\* l - длина

\*\*\* n - позиция

\*\*\* s - подстрока

\*\*\* ch - char

\*\*\* i, itter, cc - иттераторы для пробегания по строке/ам

\*\*\* code\_error - Код ошибки для вывода соответствующего сообщения

\*\*\* const\_string - Исходная строка

\*\*\* edited\_string - Измененная строка

\*/

#define N 256

//Переменные

char const\_string[N], edited\_string[N], ch;

int i, cc = 0, itter = 0;

long int code\_error;

#include <iostream>

void Error()

{

switch(code\_error)

{

case 0:

std::cout << "\nError 0 - Symbols doesn't match!\n"

<< "Please exit and try again";

break;

case 1:

std::cout << "\nError 1 - Position is out of range, maybe you entered too long word!\n"

<< "Please exit and try again";

break;

case 2:

std::cout << "\nError 2- The word longer than line, or position entered incorrect!\n"

<< "Please exit and try again";

break;

}

}

bool isAllCorrect(char\* s, int n, int l)

{

ch = s[itter];

i = n-1;

while(ch != '\0') // Пока позиция не в конце строки

{

if(

i < n+l // Находится в пределах заданных границ

)

{

if(

ch != const\_string[i++] // Символы не совпадают

)

{

code\_error = 0;

return false; // Ошибка 0

}

if(itter < l) ch = s[++itter];

}

else // Позиция за пределами заданной границы

{

code\_error = 1;

return false; // Ошибка 1

}

if(const\_string[i] == '\0' && ch != 0) // Слово длиннее строки, либо позиция указана не верно

{

code\_error = 2;

return false; // Ошибка 2

}

}

return true;

}

char\* Delete(char\* s, int n, int l)

{

if (!isAllCorrect(s, n, l)) // Ищем проблемы

{

Error(); // Если есть ошибки, то выводим его пользователю

return "\0"; // Возращаем нулевой байт в качестве указания, что что-то пошло не так

}

i = 0; // Обнуление для пробегания по исходной строке

ch = const\_string[i]; // Возвращаем на начало строки, чтобы зайти в while

while(ch != '\0') // Пока позиция не в конце строки

{

if (i < n + l - 1 && i >= n-1) i++; // Если позиция находится в удаляемой зоне, то накручиваем i, как бы пропуская этот промежуток

else

{

ch = const\_string[i++]; // Бежим по строке и берем символы

edited\_string[cc++] = ch; // Записываем символы в новую измененную строку

}

}

return edited\_string;

}

char\* DeleteNew(char\* s, int n, int l)

{

if (!isAllCorrect(s, n, l)) // Ищем проблемы

{

Error(); // Если есть ошибки, то выводим его пользователю

return "\0"; // Возращаем нулевой байт в качестве указания, что что-то пошло не так

}

i = 0; // Обнуление для пробегания по исходной строке

ch = \*(const\_string+i); // Возвращаем на начало строки, чтобы зайти в while

while(ch != '\0') // Пока позиция не в конце строки

{

if (i < n + l - 1 && i >= n-1) i++; // Если позиция находится в удаляемой зоне, то накручиваем i, как бы пропуская этот промежуток

else

{

ch = \*(const\_string+i); // Бежим по строке и берем символы

i++;

\*(edited\_string+cc) = ch; // Записываем символы в новую измененную строку

cc++;

}

}

return edited\_string;

}

int main(int argc, char const \*argv[])

{

char L[N];

int pos, len;

while(true)

{

std::cout << "\n\n\nEnter the string: " << "\n";

std::cin.getline(const\_string, N);

std::cout << "Enter substring: " << "\n";

std::cin.getline(L, N);

std::cout << "Enter position number: " << "\n";

std::cin >> pos;

if (std::cin.fail()) // если предыдущее извлечение было неудачным,

{

std::cin.clear(); // то возвращаем cin в 'обычный' режим работы

std::cin.ignore(32767,'\n'); // и удаляем значения предыдущего ввода из входного буфера

continue;

}

std::cout << "Enter the length: " << "\n";

std::cin >> len;

if (std::cin.fail()) // если предыдущее извлечение было неудачным,

{

std::cin.clear(); // то возвращаем cin в 'обычный' режим работы

std::cin.ignore(32767,'\n'); // и удаляем значения предыдущего ввода из входного буфера

}

else break;

}

std::cout << "Edited var1: " << Delete(L, pos, len) << "\n";

std::cout << "Edited var2: " << DeleteNew(L, pos, len) << "\n";

return 0;

}

3. Вывод программы через компилятор gcc

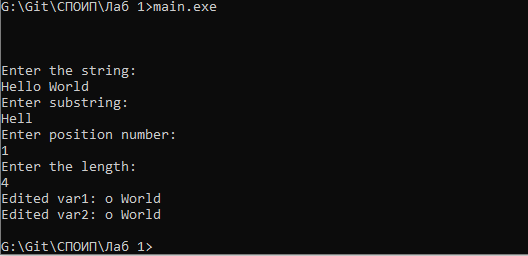


Рисунок 1 – Вывод программы

Заключение

В данной лабораторной работе были получены знания и навыки работы с символьной информацией, обработка символьных данных.

Также были получены практические навыки в работе с массивами и указателями языка C++, получен навык обеспечения функциональной модульности.

Написана программа на языке С++, реализующая алгоритм удаления подстроки из исходной строки.

Список литературы

1. Turbo Pascal. С. А. Немногин.- СПб: Издательство “Питер”, 2000.
2. Turbo Pascal: практикум/ С. А. Немногин.- СПб: Издательство “Питер”, 2001.
3. Практикум по программированию/ А. А. Чернов.- Волгоград: Учитель, 2006.
4. Приложение к журналу “Информатика и образование” № 1, 2001, № 3, 2001.
5. Немцова Т.И. Программирование на языке высокого уровня. Программирование на языке С++. М.: «Форум», 2012.

Контрольные вопросы

1.4.1В чем заключается специфика системного программного обеспечения?

1.4.2 Какие особенности языка С позволяют использовать его в качестве инструмента для системного программирования?

1.4.3 Каким образом представляются строки символов в С?

1.4.4 Что представляют собой функции?

1.4.5 С какой целью используются прототипы функций?

1.4.6 В чем разница между локальными и глобальными переменными?

1.4.7 Что такое перегрузка функции?

1.4.8 Что представляют собой указатели?

1.4.9 В чем разница между адресом, хранимым в указателе, и значением, записанным по этому адресу?

1.4.10 В чем различие между ссылкой и указателем?