**Лабораторная работа № 13. Обработка символьной информации**

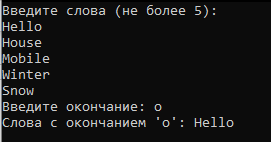
Строка символов хранится в памяти как *массив*. Каждый элемент массива содержит один символ, при этом последним символом является '**\0**'. Поэтому при объявлении массива надо указывать размерность на единицу больше, чем количество символов. Имя массива без индекса является **указателем-константой**.

|  |  |
| --- | --- |
| **Задание** | **Краткие теоретические сведения** |
| 1. Изучить способы преобразования символов, выполнив программы в правой части.  Выполнить прокрутки программ и записать условия.  Опробовать программы с различным текстом. | |  |  | | --- | --- | | **#include <iostream>**  **void main()**  **{ int num = 5;**  **char symb, new\_symbol = ' ';**  **symb = num + '0';**  **std::cout<<symb<<' ';**  **if (symb >= '0' && symb <= '9')**  **num = symb - '0';**  **std::cout<<num<<' ';**  **symb = 'b';**  **if (symb >= 'a' && symb <= 'z')**  **new\_symbol = symb - 'a' + 'A';**  **std::cout<<new\_symbol<<' ';**  **}**  Заменить четвертую и пятую строчки на операторы:  **char \*pc; pc = &symb;**  **\*pc = num + '0'; std::cout<<\*pc<<' ';**  Объяснить результат.  #include <iostream>  void main()  {  int num = 5;  char symb, new\_symbol = ' ';  char\* pc; pc = &symb;  \*pc = num + '0'; std::cout << \*pc << ' ';  if (symb >= '0' && symb <= '9')  num = symb - '0';  std::cout << num << ' ';  symb = 'b';  if (symb >= 'a' && symb <= 'z')  new\_symbol = symb - 'a' + 'A';  std::cout << new\_symbol;  }  Условие:  Преобразование числа в символ, вывод числа, преобразование строчной буквы в заглавную.  Результат:  Программа сначала выводит символ '5', который был преобразован из числа num.   Затем выводит значение num, которое остается равным 5.   Затем программа выводит символ 'B', который является заглавной версией строчной буквы 'b' | **#include <iostream>**  **void main()**  **{**  **char str[] = "Text";**  **int count = 0;**  **char \*pstr;**  **pstr = &str[0];**  **if (str)**  **while (\*pstr++)**  **++count;**  **std::cout<<count;**  **}**  В данной программе указатель может содержать нулевое значение, поэтому перед операцией разыменования его следует проверять.  Программа подсчитывает **количество символов в строке** | |
| 2. Выполнить программу, приведенную в правой части.  Объяснить принцип использования *функций стандартной библиотеки*. | Пример. Имеется адрес файла в сети, например, **http://belstu.by/p1/p2/file1.htm**.Определить имя последней папки (каталога).   |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | **#include <cstring>**  **#include <iostream>**  **using namespace std;**  **void main()**  **{**  **char str[50], s[20];**  **int i, j, i1, i2;**  **cout<<"Input string:";**  **cin>> str;**  **for(i = strlen(str); i >= 0; i--)**  **if (str[i] == '\/') break;**  **i2 = i;**  **for (j = (i2 - 1); j >= 0; j--)**  **if (str[j] == '\/') break;**  **i1 = j;**  **if (i2 == i1)**  **printf(" '\/' ");**  **else**  **{ strncpy\_s(s, &str[i1+1], i2-i1-1);**  **s[i2 - i1 - 1] = 0;**  **cout<<"katalog:"<< s<<"\n";**  **}**  **}** | |  |  | | --- | --- | | Функция | Краткое описание функции | | **strcmp** | **int strcmp(const char \*str1, const char \*str2);**  Сравнивает строки str1 и str2. Если str1 < str2, то результат <0, если str1 = str2, то результат = 0, если str1 > str2, то результат>0. | | **strcpy** | **char\* strcpy(char\*s1, const char \*s2);**  Копирует байты из строки s1 в строку s2 | | **strlen** | **int strlen (const char \*str);**  Вычисляет длину строки str | | **strncat** | **char \*strncat(char \*s1, const char \*s2, int kol);**  Приписывает kol символов строки s1 к строке s2 | | **strncpy** | **char \*strncpy(char \*s1, const char \*s2, int kol);**  Копирует kol символов строки s2 в строку s1 | | **strnset** | **char \*strnset(char \*str, int c, int kol);**  Заменяет первые kol символов строки s1 символом с | | **atoi** | **int atoi(char \*str);**  Преобразует числовые символы в целое число | | **atof** | **float atof(char \*str);**  Преобразует числовые символы в вещественное число | | |
| 3. Изучить способы работы с символьными массивами, выполнив программу в правой части. | Пример. Из предложения удалить все символы, совпадающие с символом, введенным с клавиатуры.  **void main()**  **{ char s, t[20];**  **int i, j, n;**  **cout<<"Input string: ";**  **gets\_s(t);**  **cout<<"Input symbol: ";**  **cin>>s;**  **n = strlen(t);**  **for(i = 0; i <= n; i++)**  **if (t[i] == s)**  **for(j = i; j <= n; j++)**  **t[i] = t[i + 1];**  **for(i = 0; i < n; i++)**  **cout << t[i];**  **}** |
| 4. Выполнить программу в правой части.  Закомментировать строку **cin.ignore…**  Объяснить возникшие проблемы при выполнении программы. | **void main()**  **{ char t[3][20];**  **int i, n;**  **cout<<"Input size (<=3): ";**  **cin>>n;**  **cin.ignore(cin.rdbuf()->in\_avail());**  **for(i = 0; i < n; i++)**  **{ cout<<"Input string: ";**  **gets\_s(t[i]);**  **}**  **for(i = 0; i < n; i++)**  **puts(t[i]);**  **}**    Без cin.ignore    У потока ввода есть буфер чтения, в котором он хранит символы. При неоднократном чтении символов в буфере могут остаться ненужные символы.  Буфер очищается с помощью:  **cin.ignore(cin.rdbuf()->in\_avail());**  Функция **rdbuf()** обращается к буферу, функция **in\_avail()** определяет количество символов, которое нужно проигнорировать с использованием **ignore()**. |
| 5. В правой части записаны два варианта решения задачи. Выполнить программы и объяснить различия между ними.  Внести изменения в программу с тем, чтобы проверялось не только количество скобок, но и правильность их расстановки (первой в тексте должна быть открывающая скобка). | Пример проверки соответствия количества открывающих и закрывающих круглых скобок в строке.   |  |  | | --- | --- | | **#include <stdio.h>**  **void main()**  **{**  **char s[256];**  **int i, count;**  **puts("Enter string: ");**  **gets\_s(s);**  **for(count=i=0; s[i] != 0; i++)**  **{**  **if (s[i] == '(') count++;**  **if (s[i] == ')') count--;**  **}**  **if (!count)**  **puts("Ok\n");**  **else**  **puts("Not Ok\n");**  **}** | **#include <stdio.h>**  **void main()**  **{**  **char s[256];**  **int count;**  **char \*ps;**  **puts("Enter string: ");**  **gets\_s(s);**  **for(count = 0, ps = s; \*ps != 0; ps++)**  **{**  **if (\*ps == '(') count++;**  **if (\*ps == ')') count--;**  **}**  **if (count == 0)**  **puts("Ok\n");**  **else**  **puts("Not Ok\n");**  **}** | |

|  |  |
| --- | --- |
| **№ варианта** | **Условие задачи** |
| **11** | 1. Вывести текст, составленный из последних букв всех слов.   // Вывести текст, составленный из последних букв всех слов.  #include <iostream>  #include <cstring>  using namespace std;  int main() {  setlocale(LC\_ALL, "RU");  char str[50];  cout << "Введите строку: "; cin.getline(str, 50);  int k = strlen(str);    for (int i = 0; i <= k; i++) {  if (str[i] == ' ' || str[i] == '\0') {  cout << str[i - 1];  }  }  return 0;  }   1. Дана строка символов, состоящая из произвольного текста, слова разделены пробелами. Найти порядковый номер слова максимальной длины и номер позиции в строке, с которой оно начинается.   #include <iostream>  #include <cstring>  using namespace std;  int main() {  setlocale(LC\_ALL, "RU");  char str[50];  cout << "Введите строку: ";  cin.getline(str, 50);  char\* pstr = str;  char\* maxWordStart = nullptr;  char\* currentWordStart = nullptr;    int maxLength = 0;  int currentLength = 0;  int wordIndex = 0, maxWordIndex = 0;  int position = 0;    while (\*pstr) {  if (\*pstr != ' ' && \*pstr != '\t') {  if (currentWordStart == nullptr) {  currentWordStart = pstr;  wordIndex++;  }  currentLength++;  }  else {  if (currentWordStart) {  if (currentLength > maxLength) {  maxLength = currentLength;  maxWordStart = currentWordStart;  maxWordIndex = wordIndex;  position = currentWordStart - str;  }  currentWordStart = nullptr;  currentLength = 0;  }  }  pstr++;  }  if (currentWordStart && currentLength > maxLength) {  maxLength = currentLength;  maxWordStart = currentWordStart;  maxWordIndex = wordIndex;  position = currentWordStart - str;  }  if (maxWordStart) {  cout << "Самое длинное слово: ";  for (int i = 0; i < maxLength; i++) {  cout << \*(maxWordStart + i);  }  cout << endl;  cout << "Порядковый номер слова: " << maxWordIndex << endl;  cout << "Начальная позиция в строке: " << position << endl;  }  else {  cout << "В строке нет слов." << endl;  }  return 0;  } |

Дополнительные задания.

1. В заданной последовательности слов найти все слова, имеющие заданное окончание.



#include <iostream>

#include <string>

using namespace std;

bool hasEnding(const char\* word, const char\* ending) {

int wordLen = 0;

int endingLen = 0;

while (word[wordLen] != '\0') {

wordLen++;

}

while (ending[endingLen] != '\0') {

endingLen++;

}

if (wordLen >= endingLen) {

return (0 == strcmp(word + wordLen - endingLen, ending));

}

return false;

}

int main() {

setlocale(LC\_ALL, "RU");

const int MAX\_WORDS = 5;

const int MAX\_LENGTH = 20;

char words[MAX\_WORDS][MAX\_LENGTH];

string ending;

cout << "Введите слова (не более " << MAX\_WORDS << "): " << endl;

int count = 0;

while (count < MAX\_WORDS && cin >> words[count]) {

count++;

}

cout << "Введите окончание: ";

cin >> ending;

cout << "Слова с окончанием '" << ending << "': ";

bool found = false;

for (int i = 0; i < count; i++) {

if (hasEnding(words[i], ending.c\_str())) {

cout << words[i] << " ";

found = true;

}

}

if (!found) {

cout << "Нет слов с указанным окончанием.";

}

cout << endl;

return 0;

}

1. В имеющемся словаре найти группы слов, записанных одними и теми же буквами и отличающиеся только порядком расположения.

#include <iostream>

#include <string>

#include <algorithm>

using namespace std;

const int MAX\_WORDS = 10;

const int MAX\_LENGTH = 20;

int main() {

setlocale(LC\_ALL, "RU");

char dictionary[MAX\_WORDS][MAX\_LENGTH] = {

"hello", "elloh", "world", "dowrl", "lolhe", "may", "yam", "minsk", "ared", ""

};

bool grouped[MAX\_WORDS] = { false };

cout << "Группы слов, состоящих из одних и тех же букв:" << endl;

for (int i = 0; i < MAX\_WORDS; i++) {

if (dictionary[i][0] == '\0' || grouped[i]) continue;

string currentWord = dictionary[i];

string sortedCurrent = currentWord;

sort(sortedCurrent.begin(), sortedCurrent.end());

cout << "{ " << currentWord << " ";

for (int j = i + 1; j < MAX\_WORDS; j++) {

if (dictionary[j][0] == '\0') break;

string compareWord = dictionary[j];

string sortedCompare = compareWord;

sort(sortedCompare.begin(), sortedCompare.end());

if (sortedCurrent == sortedCompare) {

cout << compareWord << " ";

grouped[j] = true;

}

}

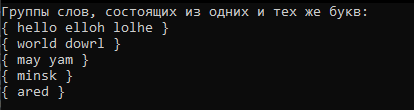
cout << "}" << endl;

grouped[i] = true;

}

return 0;

}



1. Из заданного предложения удалить те слова, которые уже встречались в предложении раньше.

#include <iostream>

#include <cstring>

#include <climits>

const int MAX\_WORDS = 100;

const int MAX\_LENGTH = 20;

using namespace std;

int main() {

setlocale(LC\_ALL, "RU");

char sentence1[256];

char sentence2[256];

char words1[MAX\_WORDS][MAX\_LENGTH];

char words2[MAX\_WORDS][MAX\_LENGTH];

int count1 = 0, count2 = 0;

cout << "Введите первое предложение: ";

cin.getline(sentence1, sizeof(sentence1));

cout << "Введите второе предложение: ";

cin.getline(sentence2, sizeof(sentence2));

char\* ptr = sentence1;

while (\*ptr != '\0' && count1 < MAX\_WORDS) {

while (\*ptr == ' ') {

ptr++;

}

if (\*ptr == '\0') {

break;

}

int length = 0;

while (ptr[length] != ' ' && ptr[length] != '\0') {

length++;

}

if (length < MAX\_LENGTH) {

for (int j = 0; j < length; j++) {

words1[count1][j] = ptr[j];

}

words1[count1][length] = '\0';

count1++;

}

ptr += length;

}

ptr = sentence2;

while (\*ptr != '\0' && count2 < MAX\_WORDS) {

while (\*ptr == ' ') {

ptr++;

}

if (\*ptr == '\0') {

break;

}

int length = 0;

while (ptr[length] != ' ' && ptr[length] != '\0') {

length++;

}

if (length < MAX\_LENGTH) {

for (int j = 0; j < length; j++) {

words2[count2][j] = ptr[j];

}

words2[count2][length] = '\0';

count2++;

}

ptr += length;

}

char\* shortestWord = nullptr;

int minLength = INT\_MAX;

for (int i = 0; i < count1; i++) {

bool found = false;

for (int j = 0; j < count2; j++) {

if (strcmp(words1[i], words2[j]) == 0) {

found = true;

break;

}

}

if (!found && strlen(words1[i]) < minLength) {

minLength = strlen(words1[i]);

shortestWord = words1[i];

}

}

if (shortestWord != nullptr) {

cout << "Самое короткое слово из первого предложения, которого нет во втором: " << shortestWord << endl;

}

else {

cout << "Все слова из первого предложения присутствуют во втором." << endl;

}

return 0;

}

