1. Добавлено в пример **17.1** (папка 17, пример 1) то, что выделено. Замените пример в папке.

// **Пример 1**. Использование стандартных функций strcpy\_s(), strcat\_s()

// в программе.

#include <iostream>

#include <cstring>

using namespace std;

const int buf = 50;

int main(){

int n;

cout<<"Vvedi max chislo simvolov v stroke == **50**: ";

cin>>n; // **ввести 50** (как размер буфера)!!!

char \* str = new char[n+1]; // выделение памяти для str

strcpy\_s(str, buf, "\*\*\*\*\*"); // копирование "\*\*\*\*\*" в str

cout << "str = " << str << endl; // \*\*\*\*\*

char \* str1 = new char[n+1]; // выделение памяти для str1

strcpy\_s(str1, buf, "55555");

cout << "str1 = " << str1 << endl; // 55555

strcpy\_s(str, buf, str1);

cout << "str = " << str << endl; // 55555

char str2[] = "77777";

cout << "str2 = " << str2 << endl; // 77777

strcpy\_s(str, buf, str2);

cout << "str = " << str << endl; // 77777

**char str3[buf];**

**cin.ignore();**

**cout<<"Vvedi stroku: ";**

**gets\_s(str3); //SSSSS**

**int len = strlen(str3)+1;**

**char\* str4 = new char[len];**

**strcpy\_s(str4, len, str3);**

**cout << "str4 = " << str4 << endl; // SSSSS**

**strcat\_s(str, buf, str4);**

**cout << "str = " << str << endl; // 777777SSSSS**

}

Следующий материал – ***дополнение к строкам***. Слова в строке разделяются не только пробелами. Здесь рассматриваются 2 функции, с помощью которых можно находить слова, выделять их. Есть и другие функции (например, strtok()).

Материал обязателен на 8-10 баллов. А так – **по желанию**.

1. **Поиск и выделение слов в строке.** Под словом будем понимать последовательность символов в строке, отличных от специальных знаков, используемых в качестве разделителей слов, например – точка, запятая, пробел, скобка и др. Между словами может любое количество *символов-разделителей*.

Для поиска слов в строке ***можно***, например, использовать функции **strspn()** и **strcspn():**

* int **strspn**(char\* str, char\* pat);– функция выполняет *поиск первого символа* строки **str***,* которого нет в строке **pat**.

Первый аргумент – строка **str**, в которой выполняется поиск, второй – строка **pat**, содержащая *символы-разделители* слов в строке. Функция просматривает строку **str** и возвращает *количество* идущих подряд символов, содержащихся в строке **pat** т.е. в **str** будет найдено *начало слова*.

Поиск всегда осуществляется, начиная с текущего адреса в строке **str**.

* int **strsсpn**(char\* str, char\* pat);*–* функциявыполняет *поиск* первого вхождения в строку **str** *любого и*з *символов* строки **pat***,* и возвращает количество символов до найденного первого вхождения.

Функция имеет те же аргументы, что и функция **strspn().** Функция просматривает строку **str** и возвращает количество подряд идущих символов, которых нет в строке **pat** т.е. в строке **str** будет найден *конец слова*. Поиск учитывает и завершающий *нуль-символ*, поэтому, если функция возвращает длину строки **str**, это значит, что ни один из символов строки **pat** не входит в состав **str**. Поиск осуществляется, начиная с текущего адреса в строке **str**.

/**/** Использование функций **strspn()** и **strspn()**.

// Поиск и выделение слов. Пошаговое выполнение**.**

…

const char str[] = "111, 22222. 55555?"; // строка

const char pat [6] = "., !?"; // символы-разделители

const char \*p = str;

p = p + **strspn**(p, pat); // первый раз функция вернёт **0**

cout<<p<<endl; //111, 22222. 55555?

int len = **strcspn**(p, pat);

cout<<"len = "<<len<<endl; // len = 3

**char\* temp1 = new char[len + 1];**

**strncpy\_s(temp1, len + 1, p, len);**

**cout << "temp1 = " << temp1 << endl; // temp1 = 111**

p = p + len;

cout<<p<<endl; //, 22222. 55555?

p = p + **strspn**(p, pat);

cout<<p<<endl; //22222. 55555?

len = **strcspn**(p, pat);

cout<<"len = "<<len<<endl; // len = 5

**char\* temp2 = new char[len + 1];**

**strncpy\_s(temp2, len + 1, p, len);**

**cout << "temp2 = " << temp2 << endl; // temp2 = 22222**

p = p + len;

cout<<p<<endl; //. 55555?

cout<<str<<endl; // 111, 22222. 55555?

…

/**/ Пример** 1**.** В строке найти количество слов, у которых первый

// и последний символы cовпадают.

#include <iostream>

using namespace std;

#include <cstring>

int **main()**{

int n;

cout<<"Vvedi max chislo simvolov v stroke:"; cin>>n;

char \* pstr = new char [n+1]; // выделение памяти для строки

const char pat[] = " ,.;:()"; // символы-разделители

char \*p = pstr, sn, sk;

int k = 0;

cout<<"vvedi stroku: ";

**cin.ignore();**

**gets\_s(pstr, n+1);**

cout<<" pstr = "; puts(pstr); cout<<endl;

do{

p = p + **strspn**(p, pat); // в ***p*** – адрес начала слова

sn = \*p; // первый символ слова

p = p + **strcspn**(p, pat); //адрес первого разделителя, т.е. адрес

// символа,следующего после последнего символа слова

sk = \*(p - 1); // последний символ слова

if(sn == sk) k++; // сравниваем символы

}while(\*p); // пока не найден конец строки

cout<<"k = "<<k<<endl;

cout << "pstr = " << pstr << endl;

cout << "p = " << p << endl;

delete [] pstr;

}

**Результат:**

Vvedi max chislo simvolov v stroke:40

vvedi stroku: 121 3456 565 7

pstr = 121 3456 565 7

k = 3

pstr = 121 3456 565 7

p =

// **Пример 2**. Дан текст из **n** строк. Отсортировать строки

// по количеству слов в строке.

#include <iostream>

using namespace std;

#include <cstring>

int kolSlov(char \*str); // прототип функции **kolSlov ()**

int **main()**{

int n, m;

cout<<"Vvedi kol-vo strok: "; cin>>n;

cout<<"Vvedi kol-vo simvolov v strokah: "; cin>>m;

char\*\* str = new char\* [n];

for(int i = 0; i < n; i++)

str[i] = new char[m+1];

**cin.ignore();**

cout<<"Vvedi "<<n<<" strok dlinoi do "<<m <<" simvolov"<<endl;

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

gets\_s(str[i], m+1);

}

cout<<endl;

cout<<"Massiv strok = \n";

for(int i = 0; i < n; i++)

cout<<"str["<<i<<"] = "<<str[i]<<endl;

int flag = 1, k = 1;

char \*x;

int k1, k2;

while(flag){

flag = 0;

for(int i = 0; i < n - k; i++){

k1 = **kolSlov**(str[i]);

k2 = kolSlov(str[i + 1]);

if(k1 > k2){

x = str[i]; str[i] = str[i+1]; str[i+1] = x;

flag = 1;

}

}

k++;

}

cout<<endl;

cout<<"Text posle perestanovki: \n";

for(int i = 0; i < n; i++)

cout<<str[i]<<endl;

cout<<endl;

for(int i = 0; i < n; i++)

delete [] str[i];

delete [] str;

}

int **kolSlov**(char \*str){ // функция подсчёта количества слов в строке

int kol = 0;

char \*temp;

temp = str;

while(\*temp){

temp = temp + strspn(temp," ,.:;");

int n = strcspn(temp," ,.:;");

kol++;

temp = temp + n;

}

return kol;

}

**Результат:**

Vvedi kol-vo strok: **3**

Vvedi kol-vo simvolov v strokah: **20**

Vvedi 3 strok dlinoi do 20 simvolov

123 456 67

34 56 78

234

Massiv strok =

str[0] = 123 456 67

str[1] = 34 56 78

str[2] = 234

Text posle perestanovki:

234

123 456 67

34 56 78