**27.03.2024 г. Символьные массивы и стандартные функции**

**Символьные массивы и стандартные функции. Символьные двумерные массивы (массивы строк). Символьные двумерные динамические массивы (массивы строк). Функция****strtok\_s(). Функции**  **strcmp(), strncmp(). Функции to\_string(), atoi().**

**Символьные массивы и стандартные функции**

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

// в программе.

#include <iostream>

#include <cstring>

using namespace std;

int **main()**{

const int buf = **80**;

int n;

cout << "Enter **max--**number simvolov in **char--array** (ravno size buf) : ";

cin >> n; // **80**

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

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

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

//-----------------------------------------------------------------------------------------------

char\* ps1 = new char[n + 1]; // выделение памяти для **ps1**

strcpy\_s(ps1, buf, "777777");

strcpy\_s(ps, buf, ps1);

cout << "ps = " << ps << endl; // **777777**

//-----------------------------------------------------------------------------------------------

char str[buf];

cin.ignore();

cout << "Enter char-array: "; gets\_s(str); // **SSSSS**

int len = strlen(str) + 1;

char\* ps2 = new char[len];

strcpy\_s(ps2, len, str);

cout << "ps2 = " << ps2 << endl; // **SSSSS**

//-----------------------------------------------------------------------------------------------

strcat\_s(ps, buf, ps2);

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

delete [] ps; delete [] ps1; delete [] ps2;

}

В программе можно использовать и другие функции стандартной библиотеки **cstring,** например **strncpy\_s(), strncat\_s(),** **strchr(), strrchr(), strcmp(), strncmp().**

// **Пример 2**. Определить, сколько раз заданная подстрока

// встречается в массиве. Использование функции **strstr().**

#include <iostream>

#include <cstring>

using namespace std;

int **main()**{

int k = 0, n, len;

cout << " Enter max length char--array n: "; cin >> n;

char\* ps = new char[n + 1];

cin.ignore();

cout << " Enter char--array : "; gets\_s(ps, n + 1);

const char\* sub = "ab"; // подстроку можно и ввести

len = strlen(sub); // длина **sub**

char \*q, \*p = ps;

while (1){

q = strstr(p, sub); // нахождение адреса подстроки **sub**

if (q != 0) k++; else break;

p = q + len; // **р** будет указывать на символ,

// след. после найденной подстроки

}

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

delete [] ps;

}

**Результат**:

Enter max length char--array n: **40**

Enter char-array : **ab**123 **ab** 456 **ab**

**k = 3**

// **Пример 3**. Удаление подстроки в символьном массиве.

// Использование стандартных функций.

#include <iostream>

#include <cstring>

using namespace std;

void subDel(char\*, int, int); // прототип **subDel()**

int **main()** {

int n;

cout << "Enter max-number simvolov in char--array: ";

cin >> n;

char\* ps = new char[n + 1];

cin.ignore();

cout << "Enter char--array: "; gets\_s(ps, n + 1);

cout << "main ps: " << ps << endl;

**subDel**(ps, 3, 4); // начиная **с ps[3]** символа, удалить 4 символa

cout << "main ps: " << ps << endl;

delete [] ps;

}

void **subDel**(char\* pstr, int nach, int kol){

cout << "subDel ps: " << pstr << endl;

int buf = strlen(pstr) + 1;

cout << " buf = " << buf << endl;

char\* temp = new char[buf];

strcpy\_s(temp, buf, pstr + nach + kol);

cout << " temp: " << temp << endl;

**pstr**[nach] = '\0';

cout << " pstr: " << pstr << endl;

strcat\_s(pstr, buf, temp);

cout << " pstr: " << pstr << endl;

}

**Результат**:

Enter max-number simvolov in char--array: 30

Enter char-array: 11134567777

**main:** ps = 11134567777

**subDel** pstr: 11134567777

buf = 12 temp: 7777 pstr: 111 pstr: 1117777

**main:** ps = 1117777

// **Пример 4**. Выделение **подстроки** в **char**-массиве. Память для

// массива и подстроки выделяется в **main(),** и её адреса передаются

// в функцию. Использование стандартных функций.

#include <iostream>

#include <cstring>

using namespace std;

void **subVid**(char\*, char\*, int, int);

int **main()**{

int n;

cout << "Enter max-number simvolov in char-array: ";

cin >> n;

char\* str = new char[n + 1];

char\* sub = new char[n + 1];

cin.ignore();

cout << "Enter char-array :"; gets\_s(str, n + 1);

cout << "main str: " << str << endl;

subVid(str, sub, 3, 4); // с **str[3]** выделение **4** символов

cout << "main sub: " << sub << endl;

cout << "main str: " << str << endl;

delete [] str; delete [] sub;

}

void **subVid**(char\* p**s**, char\* sub, int nach, int kol) {

cout << "subVid s: " << ps << endl;

int buf = strlen(ps) + 1;

cout << " buf = " << buf << endl;

strncpy\_s(**sub**, buf, ps + nach, kol);

cout << " sub: " << sub << endl;

}

**Результат**:

Enter max-number simvolov in char-array: **30**

Enter **char-array** :11134567777

**main** str: 11134567777

**subVid** ps: 11134567777

buf = 12

sub: 3456

**main** sub: 3456

**main** str: 11134567777

// **Пример 5.** В **char**-массиве найти количество слов, у которых

// первый и последний символы cовпадают.

// Использование функций **strspn(), strсspn()**.

#include <iostream>

#include <cstring>

using namespace std;

int **main()**{

int n;

cout << "Enter max-number simvolov in char-array:"; cin >> n;

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

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

char\* p = pstr, sn, sk;

cout<<"adresa pstr and p: "<<hex<<(int)pstr<<" "<<(int)p<<endl;

// cout<<hex<<(int)&pstr[0]<<endl; // или так

// cout<<hex<<(int)&p[0]<< endl;

int k = 0;

cin.ignore();

cout << "Enter char-array: "; gets\_s(pstr, n + 1);

cout << "pstr: "; puts(pstr);

cout << endl;

do {

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

cout<<" p = "<<p<< hex<<" "<<(int)p<< endl;

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; // строка не изменилась

cout<<"adres pstr: "<<hex<<(int)&pstr[0]<<endl; // адрес строки **pstr**

cout << "p: " << p << endl;

cout<<"adres p: "<<hex<<(int)&p[0]<<endl; // адрес строки p

delete [] pstr;

}

**Результат:**

Enter max-number simvolov in char-array:30

adresa pstr and p: 2c250c0 2c250c0

Enter char-array: 121, 3456 565 7

pstr: 121, 3456 565 7

p = 121, 3456 565 7 2c250c0

p = 3456 565 7 2c250c5

p = 565 7 2c250cd

p = 7 2c250d1

k = 3

pstr: 121, 3456 565 7

adres pstr: 2c250c0

p:

adres p: 2c250d2

**Символьные двумерные массивы (массивы строк)**

// **Пример 6**. Ввод/вывод массива строк.

// Перестановка строк.

#include<iostream>

#include <cstring>

using namespace std;

int **main()** {

const int n = 3, m = 30;

char str[n][m];

// Ввод массива строк

cout << "Enter " << n << " rows up\_to " << m - 1 << " simvolov" << endl;

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

gets\_s(str[i]);

// Вывод массива строк

cout << "Array rows: \n";

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

cout << str[i] << endl;

char \* ps[n]; **//** массив указателей на строки

char \* temp;

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

ps[i] = str[i];

temp = ps[0]; ps[0] = ps[1]; ps[1] = temp; **//** перестановка строк

// Вывод массива строк после перестановки

cout << "Array rows after exchange:\n";

cout << " used array--pointers:" << endl;

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

cout << ps[i] << endl;

// Вывод исходного массива строк после перестановки

cout << "Array rows after exchange:\n";

cout << " used name\_array:" << endl;

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

cout << str[i] << endl;

}

**Результат:**

Enter 3 rows up\_to **9** simvolov:

aaaaa

bbb bbbb

ccccccc

**Array rows:**

aaaaa

bbb bbbb

ccccccc

**Array rows after exchange:;**

**used array--pointers:**

bbb bbbb

aaaaa

ccccccc

**Array rows after exchange:**

**used name\_array:**

aaaaa

bbb bbbb

ccccccc

// **Пример 7**. Ввод/вывод динамического массива строк.

// Перестановка строк.

#include<iostream>

#include <cstring>

using namespace std;

int **main()**{

int n, m;

// Динамический двумерный массив **char**

cout << "Enter n --- number rows:"; cin >> n;

char \*\* pstr = new char\* [n]; // память под массив указателей

cout << "Enter m --- number simvolov in rows: "; cin >> m;

**cin.ignore();**

cout << "Enter " << n << " rows up\_to " << m << " simvolov:\n";

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

pstr[i] = new char[m + 1]; // память под i-ую строку

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

}

cout << "Array rows: \n"; // вывод массива

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

cout << pstr[i] << endl;

char\* rab = pstr[0]; // перестановка **0** и **1**-ой строк

pstr[0] = pstr[1];

pstr[1] = rab;

// Вывод массива строк

cout << "array after exchange:\n";

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

cout << pstr[i] << endl;

// Освобождение памяти, занимаемой массивом

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

delete[] pstr[i];

delete[] pstr;

}

**Результат:**

**Тот же**, что и в **Примере** **6**, но нет вывода, используя имя массива, т. к. **нет имени** у массива.

// **Пример 8**. Ввести строку, сформировать массив слов, входящих в строку.

// Определить количество слов в строке.

// Использовать в программе **стандартные функции** библиотеки <**cstring**>

// Для выделения слов использовать функцию **strtok\_s().**

#include <iostream>

#include <cstring>

using namespace std;

int **main()**{

int kol = 0, lenslova, len;

cout << "length row: "; cin >> len;

char\* str = new char[len + 1]; // память под строку

cin.ignore();

cout << "Enter row:"; gets\_s(str, len + 1); // ввод строки

cout << "str: "<<str << endl;

int buf = strlen(str);

char\*\* slova = new char\* [buf + 1]; // массив указателей

const char\* pat = ", . ;"; // разделители слов

char\* context = NULL;

char\* word = NULL;

word = **strtok\_s(**str, pat, &context); // первое слово

while (word){

cout << " word: " << word << endl;

cout << "context: " << context << endl;

if (word != NULL){

lenslova = strlen(word);

slova[kol] = new char[lenslova + 1];

strcpy\_s(slova[kol], lenslova + 1, word);

kol++;

word = strtok\_s(NULL, pat, &context);

}

}

cout << "str: "<<str << endl;

cout << " Array words: " << endl; // вывод массива

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

cout << slova[i] << endl;

delete[] str;

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

delete [] slova[i];

delete [] slova;

}

**Результат:**

**str:** 111 2222, 333 4444

word: 111

**context**: 2222, 333 4444

word: 2222

**context:** 333 4444

word: 333

**context:** 4444

word: 4444

**context:**

**str:** 111

**Array words:**

111

2222

333

4444

**Функции**  **strcmp(), strncmp()**

int **strcmp**(char\* str1, char\* str2);

Сравниваетстроки**str1** и**str2** (сравниваются коды символов)**.** Функция возвращает целое число, ***равное нулю***, если строки равны, ***меньше* *нуля***, если строка **str1** меньше строки **str2** и ***больше нуля***, если строка **str1** больше строки **str2**.

int **strncmp**(char\* str1, char\* str2, int size);

Сравнивает **заданное количество** символов строк**str1** и**str2.** Функция работает аналогично функции **strcmp(),** но сравниваются **size** символов.

#include <cstring>

int **main(){**

const char \* ps1 = "lena";

//char str[] = "lena";

const char \* ps2 = "lena";

const char \* ps3 = "Lena";

cout<<" ps1 = "<<ps1<<" ps2 = "<<ps2<<endl;

int k1 = strcmp(ps1, ps2); cout<<"k1 = "<<k1<<endl;

cout<<" ps1 = "<<ps1<<" ps3 = "<<ps3<<endl;

int k2 = strcmp(ps1, ps3); cout<<"k2 = "<<k2<<endl;

}

**Результат:**

ps1 = lena ps2 = lena

k1 = 0

ps1= lena ps3= Lena

k2 = 1

**Функции to\_string(), atoi()**

#include <cstring>

int **main(){**

int n; char st[10];

cout << "Enter a numbers: "; cin >> n;

cout << to\_string(n)<<endl;

cout << "Enter char-array: "; cin >> st;

cout << atoi(st) << endl;

}

**Результат:**

Enter a numbers : 1234

1234

Enter char - array : 4567

4567

**Д. З. на 17.04.2024**

**Постановка задания.**

**!!!** В программах использовать **только** учебный материал.

**!!!** В задачах, где нет тестовых массивов**, обязательно** включить в условие задачи.

**Ограничения:**

* не создавать для хранения цифр числа **массив;**
* не задавать числа как **символьный массив**;
* не использовать объекты класса **string**;

**Требования к программе:**

* в программе использовать динамические символьные массивы (**строки)** или динамические **массивы строк** (согласно условию задачи);
* создать и использовать одну или несколько своих функций для обработки символьных массивов (строк), массивов строк;
* в задачах **обязательно** использовать **стандартные функции ---** для выделения, удаления, замены, вставки слов (и **только в самом слове** при выборе его символов можно использовать индекс). Иначе оценка --- **не выше 5!!!;**
* использовать **меню** для выбора функций --- **по желанию**;
* результат выводить в доступном и понятном виде.

**Задачи.**

1. Ввести/вывести символьный массив (**строку**). Если в символьном массиве количество слов, которые имеют **заданную длину** и начинаются с заглавной буквы **A**, нечётное, то вывести эти слова на экран, Иначе (количество таких слов чётное) в словах массива (т.е. в самом массиве), которые имеют **заданную длину** и начинаются на букву **A**, заменить её на прописную. Например:

* **исходный массив** --- Aaaa 111 Ab5b C11 A333 Xxxx 66cc c55A

**массив-результат (на экране)** --- Aaaa Ab5b A333

* **исходный массив** --- Aaaa 111 Ab5b C11 A333 Xxxx 66cc A55A

**массив-результат** --- aaaa 111 ab5b C11 a333 Xxxx 66cc a55A

1. Ввести/вывести символьный массив (**строку**). Если количество слов в массиве чётное, то вывести на экран все слова, состоящие только из цифр. Иначе (количество слов нечётное) в словах массива (т.е. **в самом массиве**), начинающихся на цифру 5, заменить её символом ‘\*’. Например:

* **исходный массив** --- aaaa 1111 bb5b 5111 3333 5xxx 66cc 5555

**массив-результат (**на экране**)** ---1111 5111 3333 5555

* **исходный массив** --- 5aaa 111 b5b 5111 333 5xxx 555

**массив-результат --- \***aaa 111 b5b \*111 333 \*xxx \*55

1. **\***Ввести/вывести символьный массив (**строку**). Определить, сколько слов в символьном массиве заключены в круглые скобки. Вывести такие слова на экран. Если их больше одного, **удалить** первое такое слово из массива. Иначе заменить его звёздочками. Например:

* **исходный массив** --- (111) 2222 (333) (4444) 5555 66 (7777)

**массив-результат (**на экране**)** --- (111) (333) (4444) (7777)

**массив-результат (**после удаления**)** --- 2222 (333) (4444) 5555 66 (7777)

* **исходный массив** --- 111 2222 333 4444 66 (7777)

**массив-результат (**после замены**)** --- 111 2222 333 4444 66 (\*\*\*\*)

1. **\***Ввести/вывести символьный массив (строку). Определить, сколько слов в массиве максимальной длины. Вывести такие слова на экран. Если их больше одного, **удалить** первое такое слово из массива. Иначе заменить его звёздочками. Например:

* **исходный массив** --- 111 22222 333 44444 5555 66 77777

**массив-результат (**на экране**)** --- 22222 44444 77777

**массив-результат (**после удаления**)** --- 111 333 44444 5555 66 77777

* **исходный массив** --- 111 22222 333 4444 5555 66 7777

**массив-результат (**после замены**)** --- 111 \*\*\*\*\* 333 4444 5555 66 7777

1. **\*\***Ввести/вывести символьный массив (строку). Выбрать в массиве слова-палиндромы, и длина которых чётная. **Создать массив** таких слов и отсортировать его в алфавитном порядке, используя эффективную сортировку методом "**пузырька**" (т.е. без лишних просмотров элементов массива).
2. **\*\***Ввести/вывести символьный массив (**строку**). Выбрать в массиве слова, длина которых простое число. **Создать массив** таких слов и отсортировать массив в порядке возрастания длин слов, используя эффективную сортировку методом "**пузырька**" (т.е. без лишних просмотров элементов массива). Можно создать дополнительный одномерный массив длин таких слов в строке.
3. **\*\*\***Ввести/вывести массив строк. В каждой строке массива удалить слова, длина которых совпадает с заданной длиной. Отсортировать массив строк по количеству удалённых слов, используя эффективную сортировку методом "**пузырька**" (т.е. без лишних просмотров элементов массива). Можно создать дополнительный одномерный массив количеств удалённых слов в строках.
4. **\*\*\***Ввести/вывести массив строк. В каждой строке заменить слова, длина которых совпадает с заданной длиной, на звёздочки (\*\*\* \*). Отсортировать массив строк по количеству замен, используя эффективную сортировку методом "**пузырька**" (т.е. без лишних просмотров элементов массива). Можно создать дополнительный одномерный массив количеств таких слов в строках.