**13.03.2024 г. Указатели Символы Символьные массивы**

**Указатели и символы. Указатели и символьные массивы. Указатель на строку-константу. Динамические символьные массивы. Функции и символьные массивы.**

**Указатели и символы**

**Указатель** – это *переменная*, содержащая некоторый адрес. Поэтому указателю на тип **char** можно присвоить адрес символа и использовать его аналогично, например, как указатель на тип **int**.

// **Пример 1**. Использование указателя на тип **char.** Операции **&,** **\*, +, %, /.**

#include <iostream>

using namespace std;

int **main()**{

char s1 = '2', s2 = '3', s3;

// код символа **'2'** равен **50**, код символа **'3'** равен **51**

cout << s1 % s2 << ' ' << s1 / s2 << '\n'; // 50 **0**

char \* p1 = &s1; char\* p2(&s2);

cout << \*p1 << ' ' << \*p2 << endl; // 2 3

char\* p3 = &s3;

\*p3 = \*p1 + \*p2;

cout << (int)\*p3 << ' ' << \*p3 << endl; // 101 e

cout << \*p1 % \*p2 << ' ' << \*p1 / \*p2 << endl; // 50 0

(\*p1)++; cout<<\*p1<<endl; // 3

cout << \*p1 / \*p2 << ' ' << \*p1 % \*p2 << endl; // 1 0

}

**Указатели и символьные массивы**

Указателю на тип **char** можно присвоить адрес начала *символьного* массива, т. е. его имя (как и в случае числовых массивов, например, типа **int**):

char srt[20];

char \* ps; ps = str; // или **char \* ps = str;**

Указатель **ps** будет содержать адрес начала массива символов **str**. И его можно использовать при работе с массивом **str.** Доступ к элементам массива возможен и по имени массива, и по *указателю* **ps** на начало массива:

str[i] \*(str + i) ps[i] \*(ps + i)

Соответственно их адреса:

(int)&str[i] (int)(str + i) (int)&ps[i] (int)(ps + i)

Символьный массив можно вводить, а можно при объявлении сразу **проинициализировать** строкой-константой. В этом случае указателю на тип **char** также можно присвоить имя массива и далее использовать его аналогичным образом. Любой элемент символьного массива можно заменить завершающим символом строки **(**'**\0**'**),** и укоротить массив.

// **Пример 2.** Использование указателя на cимвольный массив.

#include <iostream>

#include <cstring>

using namespace std;

int **main()**{

// const int M = 50;

// char str[M];

// cout << "Enter char-array" << endl;

// gets\_s(str); **//** введём 1234567

char str[] = "1234567"; // инициализация

cout << "Using name: "; cout << str << endl; **//** 1234567

int len = strlen(str); cout << len << endl; // **7---**длина

char\* ps = str; // указатель на массив

cout << "Using pointer: "; cout << ps << endl; // 1234567

len = strlen(ps); cout << len << endl; // 7**---**длина

cout << str[1] << ' ' << ps[1] << ' ' << endl; // 2 2

cout << \*(str + 3) << ' ' << \*(ps + 3) << ' ' << endl; // 4 4

**(\*ps)++; ++(\*ps);**

cout << ps << endl; // **3**234567

cout << &str[2] <<" " << (str + 2) << endl; // 34567 34567

cout << &ps[2] <<" " << (ps + 2) << endl; // 34567 34567

str[1] = 'A'; ps[2] = 'a'; // изменяем элементы массива

cout << str << endl; // 1Aa456**7**

cout << ps << endl; // 1Aa4567

\*(ps + 4) = '\0'; // или str[4] = '\0';

cout << ps << endl; // 1Aa4

cout <<(int)str[1] << ' ' <<hex<<(int)str[1]<< endl; // 65 41

// Адреса (**по желанию**)

cout<<hex<<(int)str<<endl; // адрес массива 75fee0

cout<<hex<<(int)&str[1]<<endl; // адрес 1-**го** символа 75fee1

cout<<hex<<(int)(str+1)<<endl; // адрес 1-**го** символа 75fee1

cout<<hex<<(int)ps<<endl; // адрес массива 75fee0

cout<<hex<<(int)&ps[1]<<endl; // адрес **1-го** символа 75fee1

cout<<hex<<(int)(ps+1)<<endl; // адрес **1-го** символа 75fee1

}

Для вывода адресов символьного массива и его элементов следует выполнять *операцию приведения типа* к (int).

**Указатель на строку-константу**

В **С++** указатель на тип **char** может указывать на **строку-константу**. Но в этом случае указатель должен быть объявлен как **указатель на константу**.

// **Пример 3**. Использование указателя типа **char** на **строку-константу**.

#include <iostream>

#include <cstring>

using namespace std;

int **main**(){

**const** char \* pstr = "0123456789";

cout<<pstr<<endl; // 0123456789

cout<<\*pstr<<endl; // 0

cout<<\*(pstr+2)<<'\n'; // 2

cout<<pstr+5<<endl; // 56789

cout<<&pstr[5]<<endl; // 56789

cout<<pstr[5]<<endl; // 5

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

**const** char \* p = pstr;

cout<<p<<endl; // 0123456789

p = pstr + 5; cout<<p<<endl; // 56789

p--; cout<<p<<endl; // 456789

p = p +2; cout<<p<<endl; // 6789

}

Если объявлен **указатель на константу**, то изменить **её** нельзя, используя указатель: **\*(pstr+4) = '\*';** // **Нельзя**

**Динамические символьные массивы**

Для создания динамического символьного массива, т. е. созданного во время выполнения программы, в **С++** также используется операция **new** и **delete**:

char \* ps = new char[n];

delete [] ps;

После создания массива с ним можно работать **только** по указателю **ps** – получать, изменять значения его элементов-символов и т. п. Доступк **i**-му символу массива (т. е. элементу), адрес начала которого хранится в указателе **ps,** возможен следующими способами:

ps[i] \*(ps + i) // **i-й** символ

Соответственно их адреса:

(int)&ps[i] (int)(ps + i) // адрес **i-го** символа.

// **Пример 4**. Создание и удаление динамического символьного массива.

#include <iostream>

#include <cstring>

using namespace std;

int **main**(){

int n;

cout << " Enter size char-array: "; cin >> n; // 5

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

cin.ignore(); // очищаем поток ввода

cout << "Enter char-array: ";

gets\_s(ps, n + 1); // введём 12345

puts(ps); // 12345

cout<<ps<<endl; // 12345

cout<<&ps[1]<<' '<<(ps + 1)<<endl; // 2345 2345

cout << \*ps<<' '<<ps[1] << ' '<< \*(ps + 1) << endl; // 1 2 2

// Адреса (**по желанию**)

cout<<hex<<(int)ps<<endl; // адрес массива **347d240**

cout<<hex<<(int)&ps[1]<<' '<<(int)(ps + 1)<<endl; // 347d241 347d241

delete [] ps; // освобождение памяти

}

Для вывода адресов динамического символьного массива и его элементов необходимо также выполнять **операцию приведения типа** к (int).

// **Пример 5**. Добавление динамического **qs**-массива к концу

// динамического **ps**-массива. **Вариант\_1.**

#include <iostream>

#include <cstring>

using namespace std;

int **main()**{

int n;

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

cin.**ignore()**;

char\* qs = new char[n + 1]; // память для **qs-**массива

char\* ps = new char[2 \* n + 1]; // память для **ps-**массива

cout << "Enter ps-array: "; gets\_s(ps, n+1); // 12345

cout << "Enter qs-array: "; gets\_s(qs, n+1); // 6789

char\* pt = ps; char\* pr = qs; // используем рабочие указатели

while (\*pt != '\0') pt++; // находим конец **ps-**массива,

// в ptбудет адрес символа **'\0'**

// pt = pt + strlen(ps); // можно и так

while (\*pt++ = \*pr++); // добавление **qs-**массива

cout << " ps-array = " << ps << endl; // **ps** = 123456789

cout << " qs-array = "; puts(qs); // **qs** = 6789

delete [] ps; delete [] qs;

}

// **Пример 6**. Добавление динамического **qs**-массива к концу

// динамического **ps**-массива.

// **Вариант\_2.**  Без использования **рабочих** указателей **pt** и **pr**.

#include <iostream>

#include <cstring>

using namespace std;

int **main()**{

int n;

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

cin.**ignore()**;

char\* qs = new char[n + 1]; // память для **qs-**массива

char\* ps = new char[2 \* n + 1]; // память для **ps-**массива

cout << "Enter ps-array: "; gets\_s(ps, n+1); // 12345

cout << "Enter qs-array: "; gets\_s(qs, n+1); // 6789

while (\*ps != '\0') ps++; // находим конец **ps-**массива,

// ps = ps + strlen(ps); // можно и так

while (\*ps++ = \*qs++); // добавление **qs-**массива

**qs = qs - n;**  // возврат значений указателей

**ps = ps - 2 \* n;** // на начало массивов

cout << " ps-array = " << ps << endl; // **ps** = 123456789

cout << " qs-array = "; puts(qs); // **qs** = 6789

delete [] ps; delete [] qs;

}

**Функции и символьные массивы**

Символьный массив можно передать в функцию, используя в качестве аргумента имя массива. При этом передаётся адрес начала массива. Параметром функции в этом случае должен быть указатель типа **char**

// **Пример 7**. Создать/ввести динамический символьный массив.

// Составить и использовать функцию преобразования элементов

// массива (символов) к верхнему регистру.

#include <iostream>

#include <cstring>

using namespace std;

void **strUpper**(char\*); // прототип **strUpper()**

int **main()**{

int n;

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

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

**cin.ignore();**

cout<<"Enter char-array: "<<endl;

gets\_s(ps, n+1); // введём abCdeF

puts(ps); // abCdeF

**strUpper**(ps);

cout<<"\nRezult: "<<ps<<endl; // Rezult:ABCDEF

delete [] ps;

}

void **strUpper**(char \* str){ // функция **strUpper()**

int i = 0;

while(str[i]){

if(str[i] >= 'a' && str[i] <= 'z') str[i] = str[i] - 32;

i++;

}

}

В функцию можно передать и *строку-константу* – передаётся адрес её начала. Параметром функции в этом случае должен быть указатель типа **char** на *строку-константу*.

// **Пример 8**. Передача в функцию строки-константы.

// Параметр функции указатель типа **char** на строку-константу.

#include <iostream>

#include <cstring>

using namespace std;

void strShow(**const** char \*); // прототип функции **strShow()**

int **main()**{

int n;

**const** char \* pstr = "123456789"; // указатель на константу

strShow(pstr); // 123456789

}

void strShow(**const** char \* str){ // функция **strShow()**

while (\*str)

cout<<\*str++;

cout<<endl;

}

// **Пример 9**. Создать/ввести динамический символьный массив. Определить,

// есть ли в массиве заданный символ.

// Составить и использовать функцию поиска символа в массиве.

#include <iostream>

#include <cstring>

using namespace std;

int **poisk**(char\*, char); // прототип функции **poisk()**

int **main()**{

char \* pstr, simv;

int p, n;

cout<<" Enter **size** char-array: "; cin>>n;

cin.ignore();

pstr = new char[n+1];

cout<<" Enter char-array:"; gets\_s(pstr, n); // **abcdef**

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

cout<<" Enter simvol: "; cin>>simv; // введём символ '**c**'

p = **poisk**(pstr, simv); // вызов **poisk()**

if(p) cout<<"Yes"<<endl; // **Yes**

else cout<<"No "<<endl;

delete [] pstr;

}

int **poisk**(char \* ps, char simv){ // функция **poisk()**

int flag = 0;

while (\*ps)

if((\*ps++) == simv){flag = 1; break;}

return flag;

}

**Д. З. на 20 03.2024**

**!!!** Подготовка к контрольной

Задачи для подготовки к контрольной работе ---- **даны 06.03.24**

На проверку сдавать не надо!!!

**Опрос** по данной теме --- **27.03**