Массивы символов в C++

Прикладное программирование

Основные понятия

- В стандарт C++ включена поддержка нескольких наборов символов. Традиционный 8-битовый набор символов называется "узкими" символами.
- Кроме того, включена поддержка 16-битовых символов, которые называются "широкими".

Основные понятия

- Для представления символьных строк в C++ не существует специального строкового типа.
- Вместо этого строки в C++ представляются как массивы элементов типа char, заканчивающиеся терминатором строки – символом с нулевым значением '\0'.

Основные понятия

- Символьные строки состоят из набора символьных констант, заключенного в двойные кавычки:
- ”Это строка символов...”

Основные понятия

Набор констант, применяющихся в C++ в качестве символов.

- прописная буква от ’A’ до ’Z’ , от ’А’ до ’Я’;
- строчная буква от ’a’ до ’z’ , от ’а’ до ’я’;
- цифра от ’0’ до ’9’;

Основные понятия

- Набор констант, применяющихся в C++ в качестве символов:
- пустое место ’\t’ – горизонтальная табуляция код ASCII – 9;
- ’\n’ – перевод строки код ASCII – 10;
- ’\v’ – вертикальная табуляция код ASCII – 11;

Основные понятия

- Набор констант, применяющихся в C++ в качестве символов:
- символы пунктуации ! ” # $ % & ’ () * + , – . / : ;< = > ? @ [\ ] ^ _ { | } ~ ;
- управляющий символ все символы с кодами от 0 до 1F и символ с кодом 7F

Основные понятия

- Набор констант, применяющихся в C++ в качестве символов:
- пробел символ пробела код ASCII 32
- шестнадцатеричная цифра от ’0’ до ’9’ , от ’A’ до ’F’ , от ’a ’ до ’f’

Основные понятия

- При объявлении строкового массива необходимо принимать во внимание наличие терминатора в конце строки, отводя тем самым под строку на один байт больше:
- char buffer [10] ; // объявление строки размером 10 символов, включая терминатор.

Основные понятия

- Строковый массив может при объявлении инициализироваться начальным значением.
- char Wednesday [ ] = ”Среда”; // объявление и инициализация строки
- char Wednesday [ ] = {’C’, ’p’, ’e’, ’д’, ’a’, ’\0’} ; // что равносильно

Ввод строки

- В качестве оператора ввода при работе со строками вместо оператора записи в поток >> лучше использовать функцию getline ( ), так как потоковый оператор ввода игнорирует вводимые пробелы.

Ввод строки

- Синтаксис функции getline ( ) имеет вид:
- cin.getline(char *s, int n);
- Функция getline ( ) принимает два обязательных параметра: первый аргумент s указывает на строку, в которую осуществляется ввод, а второй параметр n – число символов, подлежащих вводу.

Ввод строки

- char S [6] ; // объявляет и инициализирует строку длиной в 5 символов
- cout << ”Input string:” ; // выводит на экран приглашение
- cin.getline (S , 6 , ’.’) ; // ввод строки длиной не более 5 символов, завершается точкой
- cout <<”You string: ”<< S <<’\n’ ; // выводит строку на экран

ОПРЕДЕЛЕНИЕ ДЛИНЫ СТРОК

- Для выяснения информации о длине строки в байтах в заголовочном файле string.h описана функция strlen ( ).
- Синтаксис этой функции имеет вид:
- size_t strlen (const char *s) ;

ОПРЕДЕЛЕНИЕ ДЛИНЫ СТРОК

- Данная функция в качестве единственного параметра принимает указатель на начало строки string, вычисляет количество символов строки и возвращает полученное беззнаковое целое число типа size_t.
- Функция strlen ( ) возвращает значение на единицу меньше, чем отводится под массив по причине резервирования места для символа ’\0’.

ОПРЕДЕЛЕНИЕ ДЛИНЫ СТРОК

- Следующий фрагмент демонстрирует использование функции:
- strlen ( ):
- char S [ ] = ”0123456789” ; // объявляет и инициализирует строку длиной 10 символов
- cout << ”Lenght=” << strlen (S) << ’\n’ ; // выводит на экран Length = 10
- cout << ”Size =” << sizeof (S) << ’\n’ ; // выводит на экран Size = 11

ОПРЕДЕЛЕНИЕ ДЛИНЫ СТРОК

- **Функция sizeof используется при вводе строк в качестве второго параметра конструкции cin.getline ( ), что делает код более универсальным, так как не требуется явного указания числа вводимых символов.**

КОПИРОВАНИЕ СТРОК

- Функция strcpy ( ) имеет прототип:
- Char * strcpy (char* str1 , const char* str2) ;
- и выполняет побайтное копирование символов из строки, на которую указывает str2, в строку по указателю str1. Копирование прекращается только в случае достижения нуль-терминатора строки str2, поэтому перед копированием необходимо удостовериться, что длина str2 меньше или равна длине str1.

КОПИРОВАНИЕ СТРОК

- Например, следующий фрагмент копирует в строку S значение строки "String copy":
- char S [21] ; // объявляет строку длиной 20 символов
- strcpy (S , ”String copy”) ; // копирование строки "String copy" в строку S
- cout<<S<<’\n’ ; // вывод на экран строки S

КОПИРОВАНИЕ СТРОК

следующий фрагмент скопирует в str2 окончание строки str1:

```
char S1 [21] = ”String copy” ; // объявляет и инициализирует строку длиной 20 символов
char S2 [21] ; // объявляет строку длиной 20 символов
char* pS = S1 ; // объявляет указатель на строку и
// инициализирует его адресом начала строки S1
cout << pS << ’\n’ ; // выводит на экран строку "String copy"
pS += 7 ; // увеличивает адрес pS на 7 байт
cout << pS << ’\n’ ; // выводит на экран строку "copy"
strcpy ( S2 , pS ) ; // копирует строку "copy" в строку S2
cout << S2 << ’\n’; // выводит на экран строку "copy"
```

КОПИРОВАНИЕ СТРОК

- Копирование части строки. Функция strncpy
- Функция strncpy ( ) отличается от strcpy ( ) тем, что в ее параметрах добавляется еще один аргумент, указывающий количество символов, не больше которого будет скопировано. Ее синтаксис имеет вид:
- char* strncpy (char* strl , const char* str2 , size_t num) ;

КОПИРОВАНИЕ СТРОК

- В приведённом фрагменте из строки sLong в строку sShort скопировано четыре первых символа с затиранием исходного значения начала короткой строки.
- char sLong [ ] = ”0123456789” ; // объявляет и инициализирует строку длиной 10 символов
- char sShort [ ] = ”abcdef” ; // объявляет и инициализирует строку длиной 6 символов

КОПИРОВАНИЕ СТРОК

- strncpy (sShort , sLong, 4) ; // копирует строку "0123" в начало строки sShort
- cout << sShort << '\n' ; // выводит на экран строку "0123ef"

КОПИРОВАНИЕ СТРОК

- Копирование строки с выделением памяти. Функция strdup
- Функция strdup ( ) в качестве параметра получает указатель на строку-источник, осуществляет распределение памяти, копирует в отведенную область строку и возвращает указатель на начало полученной строки-копии. Синтаксис функции следующий:
- char* strdup (const char* source) ;

КОПИРОВАНИЕ СТРОК

- В следующем примере производится копирование строки, на которую указывает указатель pS1, в строку, на которую указывает указатель pS2:
- char* pS ; // объявляет указатель на строку
- pS = strdup (”File not found”) ; // выделяет память для строки длиной 14 символов и
- // инициализирует указатель pS адресом этой строки

КОПИРОВАНИЕ СТРОК

- cout << pS << ’\n’ ; // выводит на экран строку pS: "File not found"
- cout << strlen (pS) << ’\n’ ; // выводит на экран длину строки pS: "14"

Конкатенация строк

- Конкатенация (или присоединение) строк используется для образования новой строки
- символов. Для этой операции стандартная библиотека предлагает функции strcat ( ) и strncat ( ).
- Присоединение строки. Функция strcat
- Функция strcat ( ) имеет синтаксис:
- char* strcat (char* str1 , const char* str2) ;

Конкатенация строк

- В результате работы функции содержимое строки, на которую указывает str2, присоединяется к содержимому строки, на которую ссылается str1. Возвращаемый функцией указатель str1 указывает на результирующую строку. При этом размер строкового массива str1 должен быть достаточным для хранения объединенной строки.

Конкатенация строк

- В следующем примере строка S инициализируется с помощью функции копирования strcpy ( ) и дополняется подстрокой, используя функцию strcat ( ):
- char S [26] ; // объявляет строку длиной 25 символов
- strcpy (S , ”Press any key ”) ; // инициализирует строку

Конкатенация строк

- strcat (S , ”to continue”) ; // добавляет в конец строки “to continue”
- cout << S << ’\n’ ; // выводит на экран строку “Press any key to continue

Конкатенация строк

- Присоединение части строки. Функция strncat.
- Функция strncat ( ) также осуществляет конкатенацию строк, однако, присоединяет лишь указанное в
- третьем параметре количество символов (беззнаковое целое): Функция strncat ( ) имеет синтаксис:
- char* strncat (char* str1 , const char* str2 , size_t num) ;

Конкатенация строк

- Функция возвращает указатель на начало сформированной строки str1. При этом размер строкового массива str1 должна быть достаточным для хранения объединенной строки. Следующий пример производит конкатенацию строки str1 с двумя первыми символами подстроки str2:

Конкатенация строк

- char S1 [31]= ”Press any key ” ; // объявляет и инициализирует
- char S2 [31]= ”to continue” ; // две строки длиной 30 символов
- strncat (S1 , S2 , 2) ; // добавляет два первых символа строки S2 в конец строки S1
- cout << S1 << ’\n’ ; // выводит на экран строку “Press any key to”

Сравнение строк

- Из двух строк меньше та, у которой меньше код первого несовпадающего символа. Ниже приводятся функции, выполняющие посимвольное сравнение двух строк.
- Функция strcmp ( ) производит сравнение строк, различая прописные и строчные буквы и имеет синтаксис:
- int strcmp (const char* S1 , const char* S2) ;

Сравнение строк

- В качестве параметров функция получает указатели на строки, которые сравниваются. После сравнения строк S1 и S2 данная функция возвращает в результате одно из следующих значений:

✔ < 0 – если строка S1 меньше, чем S2;

✔ = 0 – если строки эквивалентны;

✔ > 0 – если строка S1 больше, чем S2.

Сравнение строк

- Следующий пример иллюстрирует работу функции strcmp ( ):
- char S1 [ ] = ”Иванов” ; // объявляет и инициализирует
- char S2 [ ] = ”Иванцов” ;
- int i = strcmp (S1 , S2) ; // объявляет переменную типа int и инициализирует её
- cout << ”i = ” << i << ’\n’ ; // выводит на экран значение переменной

Сравнение строк

- cout << S1 << ’\t’ ; // выводит на экран Иванов
- cout<<(i>0 ? ’>’ :(i<0 ? ’<’ : ’=’)) ; // выводит на экран символ ‘<’ , ’>’ или ‘=’
- cout << ’\t’ << S2 << ’\n’ ; // выводит на экран Иванцов
- В результате переменной i будет присвоено отрицательное значение, так как строка S1 меньше, чем строка S2.

Сравнение строк

- Библиотека string.h также содержит функции, которые сравнивают две строки, не различая регистра
- символов. Прототипы этих функций имеют вид:
- int stricmp (const char *S1 , const char *S2) ;
- int strcmpi (const char *S1 , const char *S2) ;
- int strncmp (const char *S1 , const char* S2 , size_t n) ;

Преобразование строк

- Элементы символьных строк могут быть преобразованы из одного регистра в другой. Для этого используются стандартные функции _strlwr и _strupr.
- Функция strlwr принимает в качестве параметра указатель на строку символов, преобразует эту строку к нижнему регистру (строчные символы) и возвращает указатель на полученную строку. Данная функция
- имеет следующий прототип: char* strlwr(char* str) ;

Преобразование строк

- Следующий фрагмент показывает применение функции strlwr:
- char S [ ] = "Error" ; // объявление и инициализация строки
- strlwr (S) ; // преобразование строки в нижний регистр
- cout << S << '\n' ; // вывод на экран "error"

Преобразование строк

- Функция strupr объявлена следующим образом:
- char* strupr (char* str) ;
- Данная функция преобразует строку символов, на которую указывает str, в прописные буквы (к верхнему регистру). В результате работы функции возвращается указатель на полученную строку.

Обращение строк

- **Функция обращения строки strrev меняет порядок следования символов на обратный (реверс строки).**
- **Данная функция имеет прототип:**
- **char* strrev (char* str)**

Обращение строк

- Следующий пример демонстрирует работу функции strrev.
- char S [ ] = "Hello" ; // объявление и инициализация строки
- cout << S << '\n' ; // вывод на экран "Hello"
- strrev (S) ; // реверс строки
- cout << S << '\n' ; // вывод на экран "olleH"

Поиск символов

- Функция нахождения символа в строке strchr имеет следующий прототип:
- char* strchr (const char* string , int c)
- Данная функция производит поиск символа c в строке string и в случае успешного поиска возвращает указатель на место первого вхождения символа в строку. Если указанный символ не найден, функция возвращает NULL. Поиск символа осуществляется с начала строки.

Поиск символов

- char S [81] ; // объявление строки из восьмидесяти символов
- char* pS ; // объявление указателя на строку
- CharToOem ("Назвался U груздем, U пеняй U на U себя" , S) ; // инициализация строки
- cout << S << '\n' ; // вывод на экран исходной строки

Поиск символов

- pS = strchr (S , 'U') ; // возвращает указатель на первый пробел
- while (pS) // до тех пор, пока указатель pS не равен NULL
- {
- pS++ ; // увеличение указателя на единицу
- cout << pS << '\n' ; // вывод на экран символов от найденного пробела до конца строки
- pS = strchr (pS , ' ') ; // поиск следующего пробела
- }

Поиск подстрок

- При необходимости поиска в одной строке последовательности символов, заданной в другом символьном массиве (подстроке), стандартная библиотека string.h предлагает воспользоваться одной из следующих функций: strstr, strtok,
- Функция strstr описана следующим образом:
- char* strstr (const char* str , const char* substr)

Поиск подстрок

- Данная функция осуществляет сканирование строки str и находит место первого вхождения подстроки substr в строку str. В случае успешного поиска функция strstr возвращает указатель на первый символ строки str, начиная с которого следует точное совпадение части str обязательно со всей лексемой substr.
- Если подстрока substr не найдена в str, возвращается NULL.

Поиск подстрок

- Следующий пример показывает использование функции strstr.
- char S1 [81] ; // объявление строки
- CharToOem ("Производится поиск элемента" , S) ; // инициализация строки
- char S2 [81] ; // объявление строки
- CharToOem ("поиск" , S) ; // инициализация строки

Поиск подстрок

- char* pS ; // объявление указателя на строку
- pS = strstr (S1 , S2) ; // инициализация указателя на строку
- cout << pS << '\n' ; // вывод на экран "поиск элемента"
- На экран будет выведено "поиск элемента", так как подстрока, содержащаяся в S2, находится внутри строки S1 и функция strstr установит указатель pS на соответствующий элемент символьного массива S1.

Поиск подстрок

- **Функция strtok имеет синтаксис:**
- **char* strtok (char* str , const char* delim)**
- **Эта функция выполняет поиск в строке str подстроки, обрамленной с обеих сторон любым символом-разделителем из строки delim. В случае успешного поиска данная функция обрезает строку str, помещая символ '\0' в месте, где заканчивается найденная подстрока.**

Поиск подстрок

- Предположим, необходимо разбить предложение, имеющееся в строковом массиве, по словам и вывести каждое из них на экран.
- char S [81] ; // объявление строки
- char* pS ; // объявление указателя на строку
- char* Del="U.?!,"; // объявление и инициализация указателя на строку, содержащую набор разделителей

Поиск подстрок

- CharToOem ("Назвался груздем, пеняй на сябя!",S) ; // преобразование строки
- pS = strtok (S , Del) ; // инициализация указателя на строку адресом первого слова
- if (pS) cout << pS << '\n' ; // если указатель существует, выводит на экран первое слово
- while (pS) // до тех пор, пока указатель существует

Поиск подстрок

- {
- pS=strtok (NULL , Del) ; // указатель получает адрес очередного слова
- if (pS) cout << pS << '\n' ; // если указатель существует, выводит на экран очередное
- слово
- }

Поиск подстрок

- В данной программе объявляется подлежащая анализу строка S, подстрока Del, содержащая набор разделителей (Delimiters), и указатель на символьный тип данных pS. Вызов функции strtok (S , Del) сканирует строку S и как только в ней встретится любой символ, входящий в подстроку Delimiters (в данном случае это символ пробела), указатель pS станет ссылаться на начало исходной строки до найденного символа.

Поиск подстрок

- То есть pS будет содержать:
- *pS = "Назвался" ;
- Благодаря тому, что функция strtok помещает в найденном месте нуль-терминатор ('\0'), исходная строка модифицируется. Таким образом, массив символов S примет значение:
- "груздем, пеняй на сябя!"

Поиск подстрок

- Осуществив проверку указателя pS на существование в операторе if (pS), найденное слово выводится на экран. Далее в цикле с помощью функции strtok находится последний нуль-терминатор строки S:
- pS = strtok (NULL , Del) ;
- что, фактически, соответствует локализации следующего слова предложения, и найденная последовательность символов выводится на экран.

Преобразование строки в число

- Функции atoi и atol
- Синтаксис функций atoi и atol имеет вид:
- int atoi (const char* ptr) ;
- int atol (const char* ptr) ;
- Эти функции преобразуют ASCIIZ-строку символов, на которую указывает ptr, в число типа int и long соответственно. Работа этих функций в 32-разрядной модели памяти не отличается.

Преобразование строки в число

- Если преобразуемое число превышает диапазон значений типа int, функция возвратит непредсказуемое значение. Например:
- char S [41] = "400000" ; // объявление и инициализация строки
- int n ; // объявление переменной типа int
- n = atoi (S) ; // инициализация переменной значением преобразованной строки

Преобразование строки в число

- cout << "n=" << n << '\n' ; // выводит на экран “400000”
- strcat (S , "0000") ; // добавляет в конец строки четыре нуля
- long m; // объявление переменной типа long
- m = atoll (S) ; // инициализация переменной значением преобразованной строки
- cout << "m=" << m << '\n' ; // выводит на экран “-294967296”

Преобразование строки в число

- Функция atof, определенная как
- double atof(const char* ptr)
- выполняет преобразование ASCIIZ-строки в число с плавающей точкой типа double. Строка символов должна быть представлена с учетом формата:
- [пробелы][знак][цифры][.][цифры][e|E [знак]цифры],
- Где пробелы – последовательность пробелов или табуляторов;

Преобразование строки в число

- знак – символ '+' или '–' ;
- цифры – десятичные цифры;
- e|E – символ показателя степени.
- Преобразование символов прекращается, как только найден первый неконвертируемый символ или достигнут конец строки.
- Пример использования функции atof:

Преобразование строки в число

- char S [41] = "12345e66" ; // объявление и инициализация строки
- double d ; // объявление переменной типа double
- d = atof (S) ; // инициализация переменной значением преобразованной строки
- cout << "d=" << d << '\n' ; // выводит на экран "d=1.2345e+070"

Преобразование числа в строку

- **Функции _itoa и _ltoa**
- **Функции обратного преобразования itoa и _ltoa производят конвертирование чисел типа int и long соответственно в строку символов. Они имеёт следующий синтаксис:**

char *_ltoa (long value , char *string , int radix);

char *_itoa (int value , char *string , int radix) ;

Преобразование числа в строку

- Данные функции принимают в качестве аргумента число value и преобразуют его в строку string с учетом основания системы счисления, представленной в переменной radix. Следующий фрагмент программы преобразует целое число в строку, используя десятеричную систему счисления:

Преобразование числа в строку

- int numb = 98765 ;
- char S [10] ;
- _itoa (numb , S , 10) ;
- cout << numb << '\n' << S ;