В.Л. Тарасов

Лекции по программированию на C++

Лекция 6

Символы и строки

6.1. Символы

Согласно стандарту языка C++, принятому в 2011 г., для работы с символами можно использовать тип `char`, размером в 1 байт, тип `char16_t`, размером 2 байта, тип `char32_t`, размером 4 байта и тип `wchar_t`, размер которого определяется при реализации. Типы `char` и `wchar_t` были предусмотрены уже стандартом 1999 г.

Рассмотрим тип `char`. Как уже говорилось, это целый тип, с небольшим диапазоном значений, который можно использовать везде, где допустимо использовать целые.

Символьные константы

В тексте программ могут использоваться *символьные константы*, заключаемые в одиночные апострофы, например: `'g'`, `'F'`, `'Q'` – латинские буквы, `'ц'`, `'г'`, `'ж'` – русские буквы, `'0'`, `'1'`, `'9'` – цифры, `'.'`, `','`, `':'` – знаки препинания. Эти константы имеют тип `char`, а их числовые значения равны *кодам символов в кодовой таблице*. Символьные константы могут использоваться в любых выражениях, где допустимо вхождение других целых типов.

Так как под тип `char` отводится один байт памяти, состоящий из 8 двоичных разрядов, всего существует $2^8 = 256$ символов. На клавиатуре компьютера есть клавиши только для части символов, входящих в кодовую таблицу, на экране же дисплея можно показать большинство символов, так как изображение создается из отдельных точек. Символы, генерируемые клавишами **`Backspace`**, **`Tab`**, **`Esc`** и некоторыми другими не имеют изображения, так как за этими символами закреплены определенные *действия*, например, вывод на экран символа *«табуляция»* приводит к перемещению курсора в следующую позицию табуляции. Несколько символов в языке C++ имеют *служебное* назначение, например, *«двойная кавычка»*, *«апостроф»*, *«обратная наклонная черта»*.

Служебные символы и символы, у которых нет графического образа, имеют в C++ специальное представление, иногда называемое *эскейп-последовательностью*. Данное представление начинается с обратной наклонной черты (\). Специальные символы, представляемые эскейп-последовательностями, перечислены в табл. 6.1.

Таблица 6.1. Представление символов

Обозначение	Название символа	Обозначение	Название символа
\a	сигнал звонок	\?	знак вопроса
\b	возврат на шаг	\'	апостроф
\f	перевод страницы	\"	двойная кавычка
\n	новая строка	\v	вертикальная табуляция
\r	возврат каретки	\t	горизонтальная табуляция
\\	обратная наклонная черта	\ooo	восьмеричный код
		\xhh	шестнадцатеричный код

В виде эсекйп-последовательности можно представлять символы, используя их коды, определяемые кодовой таблицей. Код символа задается в восьмеричной или шестнадцатеричной системе счисления. В таблице 6.1 обозначено: ooo – одна, две или три *восьмеричные цифры*, hh – одна или две *шестнадцатеричные цифры*. Компилятор, встретив в тексте программы указанные последовательности, рассматривает их как один соответствующий символ. Например, цифру «нуль» можно представить в программе тремя способами: '0', '\60', '\x30', так как цифра «нуль» кодируется десятичным числом 48_{10}, которое в восьмеричной системе записывается как 60_8, а в шестнадцатеричной как 30_{16}.

При выводе значений типа char оператор << создает на экране изображение соответствующего символа. При необходимости вывести код символа, его следует преобразовать к типу int с помощью конструкции int(выр), где выр – выражение типа char. Сказанное иллюстрируется приводимой далее программой.

Программа 6.1. Представления символов

В программе выводится значение выражения int('A'), которое равно коду латинской буквы A, а также изображение данной буквы с использование трех представлений: 'A', '\x41', '\101' и как значение символьной переменной c.

```
// Файл RepresentationChars.cpp
// Использование символьных констант и переменных

#include <iostream>
#include <clocale>
#include <cstdlib>
```

```
using namespace std;

int main()

{
    setlocale(LC_ALL, "Russian");
    char c = 'A';            // Переменная с инициализируется латинской A
    cout << "Код: " << int('A') << ", символ: " << 'A'
         << ',' << '\x41' << ',' << '\101' << ','
         << c << ',' << char(65)  << endl;
    system("pause");
    return 0;
}
```

Данная программа выводит:

```
Код: 65, символ: A, A, A, A, A
```

Таким образом, константы 'A', '\x41', '\101' имеют одинаковое числовое значение 65, равное коду латинской A в кодовой таблице.

Ввод и вывод символов

Оператор ввода >> пропускает пробельные символы, которые обычно используются для разделения вводимых данных, поэтому этот оператор неудобен для ввода одиночных символов, так как пробельные символы будут пропускаться.

Любые символы может вводить функция

```
int get();
```

которая извлекает из входного потока очередной символ и возвращает его код. При вводе из стандартного входного потока `cin` вызов данной функции должен иметь вид: `cin.get()`, например,

```
int c;
c = cin.get();
```

Проверить возможность чтения из потока можно с помощью функции

```
bool eof(),
```

которая возвращает `true` (или 1), если достигнут конец входного потока. Такая ситуация возникает после первой неудачной попытки чтения из потока. Если не достигнут конец входного потока, `eof()` возвращает `false` (или 0). Для входного потока `cin` вызов данной функции имеет вид `cin.eof()`.

С помощью клавиатуры признак конца входного потока формируется нажатием двух клавиш **Ctrl+Z**.

Для вывода одиночного символа можно использовать функцию

`put(int c)`,

которая выводит в поток свой аргумент. При выводе в стандартный выходной поток `cout` символа `c` вызов данной функции должен иметь вид `cout.put(c)`.

В программе 6.2 демонстрируется использование данных функций.

Программа 6.2. Печать текста по словам

Приводимая здесь программа читает вводимый текст, пока не будет достигнут конец входного потока, и каждое слово текста выводит на отдельной строке. Разделителями слов считаются пробельные символы. В программе циклически повторяются действия:

1) пропустить пробельные символы;
2) прочитать и вывести символы слова;
3) перейти на новую строку выходного потока.

```
// Файл TextWords.cpp

#include <iostream>
#include <cstdlib>
using namespace std;

// Каждое слово вводимого текста выводится  на отдельной строке

int main()
{
    char c;                                 // Здесь будут читаемые символы
    while( !cin.eof() ){                    // Пока не исчерпан входной поток,
        c = cin.get();                      // читаем символ из потока
// Пропуск пробельных символов
        while(c == ' ' || c == '\n' || c == '\t') // Пока с – пробельный
            c = cin.get();                  // символ, читаем следующий

// Чтение и вывод символов слова

        while(!cin.eof() && c != ' ' && c != '\n' && c != '\t'){ // Пока с
                                                                 // не пробел
            cout.put(c);                        // Вывод символа
            c = cin.get();                      // Чтение следующего символа
        }
        cout.put('\n');  // Переход к новой строке, когда закончилось слово
    }
    system("pause");
    return 0;
}
```

Для размещения текущего символа использована переменная `c` типа `char`.

Внешний цикл `while(!cin.eof()){…}` выполняется, если во входном потоке есть символы, что проверяется с помощью вызова функции

cin.eof(). Знак ! обозначает оператор логического отрицания, поэтому, если конец входного потока не достигнут, eof() возвращает false (0), отрицание которой даст true (1), и цикл будет выполняться.

Внутри внешнего цикла сначала читается очередной символ. Первый из двух внутренних циклов while предназначен для пропуска пробельных символов. Пробельными символами считаются собственно пробел ' ', табуляция '\t' и новая строка '\n'. Используется оператор == сравнения на равенство, результаты сравнения соединяются логическим оператором ИЛИ, обозначаемым ||.

Второй внутренний цикл выполняется, пока очередной символ *не является* пробельным и *не достигнут конец входного потока*. Используется оператор сравнения на неравенство != и логический оператор И, обозначаемый &&.

Далее приводится пример работы программы.

```
Когда б имел златые горы^Z
Когда
б
имел
златые
горы
```

Для ввода признака конца входного потока нажимается комбинация Ctrl+Z, что приводит к появлению на экране ^Z.

6.2. Символьные массивы как строки

Строки символов широко используются в программировании, так как общение пользователя с программой ведется, в основном, с помощью текстовых сообщений, состоящих из отдельных строк. Для работы со строками в языках Си и C++ можно использовать массивы символов, завершаемые нулевым байтом. Нулевой байт считается концом строки. Символ с нулевым кодом можно изобразить как '\0'. Например, массив s, определенный инструкцией:

```
char s[100];
```

может содержать строку длиной от 0 до 99 символов, так как из 100 байтов, выделенных под массив, один байт занимает признак конца строки '\0'. Данный символ может занимать любую позицию в массиве, поэтому в массиве фиксированного размера могут располагаться строки различной длины. Если s[0] == '\0', то массив s содержит строку нулевой длины, иначе говоря, пустую строку.

Строковые константы

Строковая константа представляется в тексте программы последовательностью произвольных символов, заключенной в двойные кавычки. Под строковую константу в памяти выделяется массив, в котором располагаются все символы строки и признак конца строки, например, для строки "Здравствуй, Мир!" сформируется массив:

З	д	р	а	в	с	т	в	у	й	,		М	и	р	!	\0

Ввод и вывод строк

Для вывода строк можно использовать оператор <<.

При вводе оператором >> в строку включаются все символы до первого пробела. Ввод строк, содержащих любые символы, включая пробелы, обеспечивает функция:

```
int getline(char s[], int n),
```

которая читает из входного потока не более n – 1 символа до поступления символа новой строки '\n' или достижения конца файла. Символ новой строки вводится с клавиатуры нажатием Enter. Прочитанный символ новой строки '\n' не включается в массив s. При вводе из потока cin вызов функции getline() должен иметь вид:

```
cin.getline(s, n);
```

Средства работы со строками

Существует библиотека функций для работы со строками с заголовочным файлом cstring. Опишем некоторые из этих функций. В приводимых описаниях используется обозначение size_t для типа unsigned int. Данное обозначение вводится инструкцией:

```
typedef unsigned int  size_t;  // Новое название типа unsigned int
```

которая имеется в стандартных заголовочных файлах. Оператор typedef вводит новое название для типа данных, то есть там, где стоит size_t, подразумевается unsigned int. Тип size_t используется для неотрицательных величин, например, для размеров объектов.

Функция

```
size_t strlen(const char s[]);
```

возвращает длину строки s, завершающий символ '\0' не учитывается. Модификатор const запрещает изменение массива s внутри функции. Пример использования функции strlen():

```
int n;
n = strlen("Здравствуй");        // n = 10
```

Функция

```
char* strcpy(char dest[], const char src[]);
```

копирует строку `src` в строку `dest` и возвращает *указатель* на копию `dest`. О том, что такое указатель будет говориться позже, пока же это понятие не будет использоваться.

Функция

```
int strcmp(const char s1[], const char s2[]);
```

посимвольно сравнивает строки `s1` и `s2`, причем сравниваются коды символов. Возвращается значение `< 0`, если первый несовпадающий символ `s1` имеет код меньше кода соответствующего символа `s2`; значение `> 0`, если первый несовпадающий символ `s1` имеет больший код, чем соответствующий символ `s2` и `0` при полном совпадении строк. Например,

```
int n;
n = strcmp("Мама", "Папа");  // n < 0
```

Программа 6.3. Удаление символа из строки

Требуется удалить из строки все вхождения некоторого символа. Пусть, например, из строки `"На дворе трава, на траве дрова."` нужно удалить символ ‘`р`’. Должна получиться строка: `"На двое тава, на таве дова."`.

Алгоритм состоит в том, что перебираются все символы строки и копируются в тот же массив, который занимает строка, за исключением символа, который следует удалить. После прохода всей строки после последнего скопированного символа ставится признак конца строки `'\0'`.

```
#include <iostream>
#include <clocale>
#include <cstring>
using namespace std;

// delchar: удаляет символ c из строки s.
// Возвращает длину строки
int delchar(char s[], char c)
{
    int i, j;
    for(i = 0, j = 0; s[i] != '\0'; ++i)  // Перебор всех символов строки
        if(s[i] != c){                    // Если символ s[i] не удаляемый,
            s[j] = s[i];          // он копируется на новое место строки
            ++j;                  // Счетчик оставленных в строке символов
```

```
        }
    s[j] = '\0';              // Фиксируем новый конец строки
    return j;                 // Возвращаем количество символов в строке
}

int main()
{
    const int N = 200;                // Размер массива для строки
    char st[N];                       // Массив для строки
    setlocale(LC_ALL, "Russian");
    cout << "Введите строку:\n";
    cin.getline(st, N);                   // Ввод строки в массив st
    int n = strlen(st);                   // Находим длину строки
    cout << "Длина введенной строки: " << n << endl;
    cout << "Введите удаляемый символ ";
    int dc = cin.get();                   // Ввод символа
    n = delchar(st, dc);     // Вызов функции, удаляющей символ из строки

    cout << "Обработанная строка:\n";
    setlocale(LC_ALL, ".866");        // Настройка на кодовую страницу 866
    cout << st << endl;
    setlocale(LC_ALL, "Russian");
    cout << "ее длина: " << n << endl;
    system("pause");
    return 0;
}
```

При вводе строки с клавиатуры могут использоваться русские буквы. В окне выполнения программы по умолчанию используется кодовая страница 866, которая применялась в свое время для русификации программ для операционной системы DOS. Поэтому для правильного вывода русских букв, вводимых в программу с клавиатуры, устанавливается соответствующая кодовая страница:

```
setlocale(LC_ALL, ".866");
```

Далее приведен пример работы программы:

```
Введите строку:
Карл у Клары украл кораллы
Длина введенной строки: 26
Введите удаляемый символ л
Обработанная строка:
Кар у Кары укра кораы
ее длина: 21
```

Программа 6.4. Реверсирование строк

Реверсирование – это перестановка символов строки в обратном порядке. Пусть следует переставить символы в строке `s = "abcdefghi"`. Схема расположения строки `s` в памяти показана на рис.6.1.

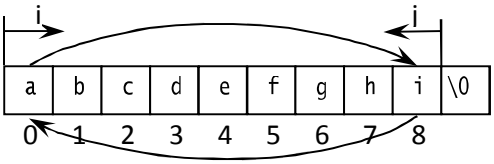

Рис. 6.1. Реверсирование строки

Алгоритм реверсирования состоит в следующем. Вводятся два счетчика: i с начальным значением, равным 0, то есть номеру первого символа строки, и j с начальным значением, равным номеру последнего символа строки. Соответствующие символы меняются местами. Далее счетчик i увеличивается на 1, а j – уменьшается на 1 и меняются местами символы s[i] и s[j]. Так продолжается до тех пор, пока счетчики не «встретятся» на середине строки. Заметим, что средний символ 'e' останется на своем месте.

```
// Файл Revers_String.cpp

#include <iostream>
#include <clocale>
#include <cstring>
#include <cstdlib>
using namespace std;

// revers: перестановка символов s в обратном порядке

void revers(char s[])
{
    int i, j;            // i - номер первого, второго,...,символа
                         // j - номер последнего, предпоследнего,...,символа
    char tmp;            // Промежуточная переменная для обмена символов
    for(i = 0, j = strlen(s) - 1; i < j; i++, j--){
        tmp = s[i]; s[i] = s[j]; s[j] = tmp;
    }
}

int main()
{
    setlocale(LC_ALL, "Russian");
    const int MAX = 100;              // Размер строк
    char orig_s[MAX];                 // Массив под исходную строку
    char rev_s[MAX];                  // Массив под реверсированную строку
    cout << "Введите строку: ";
    cin.getline(orig_s, MAX);         // Ввод строки
    strcpy(rev_s, orig_s);   // Исходную строку orig_s копируем в rev_s
    revers(rev_s);                    // Реверсирование строки rev_s
    cout << "Реверсированная строка: ";
    cout << rev_s << endl;            // Вывод реверсированной строки
    int rez = strcmp(orig_s, rev_s); // Сравнение исходной и реверсир. строк
    // Вывод исходной и реверсированной строк
```

```
    char c;                          // Символ – результат сравнения строк
    if(rez < 0)
        c = '<';
    else if(rez == 0)
        c = '=';
    else
        c = '>';
    cout << orig_s << c << rev_s;
    cout << "\nРезультат сравнения строк rez = " << rez << "\n";
    system("pause");
    return 0;
}
```

В функции `reveres()` использован цикл с заголовком:

```
for(i = 0, j = strlen(s) - 1; i < j; i++, j--)
```

Здесь применен оператор *запятая* (,), чтобы объединить в *одно* выражение *два* выражения присваивания: `i = 0` и `j = strlen(s) - 1`. Тем же оператором запятая в одно выражение объединяются выражения `i++` и `j--`.

Присваиванием `j = strlen(s) - 1` переменная `j` получает значение номера последнего символа строки, который на единицу меньше длины строки, так как нумерация элементов массива начинается с нуля.

Обмен начальных и конечных символов производится в цикле `for`, в котором `i` увеличивается, а `j` уменьшается до тех пор, пока их значения не сравняются, то есть пока не будет достигнута середина строки.

Далее приводятся результаты трех запусков программы.

```
Введите строку: abcdefghi
Реверсированная строка: ihgfedcba
abcdefghi<ihgfedcba
Результат сравнения строк rez = -1

Введите строку: 54321
Реверсированная строка: 12345
54321>12345
Результат сравнения строк rez = 1

Введите строку: kazak
Реверсированная строка: kazak
kazak=kazak
Результат сравнения строк rez = 0
```

6.3. Стандартный тип string

Стандарт языка C++ предусматривает тип данных `string` для работы со строками символов, определенный в заголовке с таким же именем `string`. Этот тип облегчает и делает более безопасной работу со строками. При работе со строками как с массивами символов

программист должен следить за размером памяти, выделяемой под строки, не должен забывать об установке символа конца строки. Тип string сам управляет памятью. Кроме того, можно использовать оператор присваивания (=), оператор (+) для объединения строк и другие.

Программа 6.5. Использование типа string

В программе показаны несколько приемов использования типа string. Смысл программы ясен из комментариев.

```
// Файл UseString.cpp

#include <iostream>
#include <clocale>
#include <cstdlib>
#include <string>                 // Для доступа к классу string
using namespace std;

int main()                        // Определение и присваивание для строк
{
    setlocale(LC_ALL, "Russian");// Настройка консоли для вывода по-русски
    string s1("Рыба");                    // Создание и
    string s2 = "Мясо";                   // инициализация строк
    string s3;                            // Создание пустой строки
    s3 =  s1;                             // Присваивание строк
    cout << "s3 = " <<  s3 << endl;       // Вывод строки
    s3 = "Ни " + s1 + " ни ";             // Склеивание строк
    cout <<  "s3 = " << s3 << endl;
    s3 += s2;                             // Добавление и присваивание строк
    cout <<  "s3 = " << s3 << endl;
    s1.swap( s2 );                        // Перестановка строк
    cout << s1 << " не " << s2 << endl;
    string name;
    cout << "Длина name = " << name.length() << endl;
    cout << "Как Вас зовут? ";
    cin >> name;
    cout << "Здравствуйте, ";
    setlocale(LC_ALL, ".866");   // Для вывода ранее введенных русских букв
    cout << name << endl;
    setlocale(LC_ALL, "Russian");
    cout << "Длина Вашего имени " << name.length() << " букв\n";
    system("pause");
    return 0;
}
```

Для обмена значений двух строк использован метод swap(), поддерживаемый типом string.

Программа выводит:

```
s3 = Рыба
s3 = Ни Рыба ни
```

```
s3 = Ни Рыба ни Мясо
Мясо не Рыба
Длина name = 0
Как Вас зовут? Владимир
Здравствуйте, Владимир
Длина Вашего имени 8 букв
```