Строки

Строки

C++ не содержит стандартного типа данных «строка».

Вместо этого он поддерживает массивы символов, завершаемые нуль-символом.

Библиотека содержит функции для работы с такими массивами, унаследованные от C и описанные в заголовочном файле <string.h> (<cstring>).

Они позволяют достичь высокой эффективности, но весьма неудобны и небезопасны в использовании, поскольку выход за границы строки не проверяется.

Строки

Тип данных string стандартной библиотеки лишен этих недостатков, но может проигрывать массивам символов в эффективности.

Основные действия со строками выполняются в нем с помощью операций и методов, а длина строки изменяется динамически в соответствии с потребностями. Для использования класса необходимо подключить к программе заголовочный файл <string>.

Пример:

```
#include <cstring>
#include <string>
#include <iostream>
using namespace std;
int main (){
   char c1[80], c2[80], c3[80]; // Строки с завершающим нулем
   string s1, s2, s3;
   // Присваивание строк
   strcpy(c1, "old string one");

    s1 = "new string one";
    s2 = s1;
    // Конкатенация строк
    strcpy(c3, c1);
    strcpy(c3, c2);
    s3 = s1 + s2;
    // Сравнение строк
    if (strcmp(c2, c3) < 0 ) cout << c2;
    else cout << c3;
    if (s2 < s3) cout << s2;
    else cout << s3;
 }
```

Строки

Как видно из примера, выполнение любых действий со строками старого стиля требует использования функций и менее наглядно. Кроме того, необходимо проверять, достаточно ли места в строке-приемнике при копировании, то есть фактически код работы со строками старого стиля должен быть еще более длинным.

Строки типа string защищены от выхода информации за их границы и с ними можно работать так же, как с любым встроенным типом данных, то есть с помощью операций. Рассмотрим основные особенности и приемы работы со строками.

Конструкторы и присваивание строк

В классе string определено несколько **конструкторов.** Ниже в упрощенном виде приведены заголовки наиболее употребительных:

```
string();
string(const char * ) ;
string(const char *, int n);
string(string &);
```

Конструкторы и присваивание строк

Первый конструктор создает пустой объект типа string.

Второй создает объект типа string на основе строки старого стиля, третий создает объект типа string и записывает туда n символов из строки, указанной первым параметром.

Последний конструктор является конструктором копирования, который создает новый объект как копию объекта, переданного ему в качестве параметра.

Конструкторы и присваивание строк

В классе string определены три операции присваивания:

```
string& operator=(const string& str);
string& operator=(const char* s);
string& operator=(char c);
```

Конструкторы и присваивание строк

Как видно из заголовков, строке можно присваивать другую строку типа string, строку старого стиля или отдельный символ, например:

```
string s1;
string s2("Вася");
string s3(s2);
s1 = 'X';
s1 = "Вася";
s2 = s3;
```

Операции

Ниже приведены допустимые для объектов класса `string` операции:

Операция	Действие	Операция	Действие
=	присваивание	>	больше
+	конкатенация	>=	больше или равно
==	равенство	[ ]	индексация
!=	неравенство	<<	вывод
<	меньше	>>	ввод
<=	меньше или равно	+=	добавление

Операции

Синтаксис операций и их действие очевидны. Размеры строк устанавливаются автоматически так, чтобы объект мог содержать присваиваемое ему значение.
Надо отметить, что для строк типа string не соблюдается соответствие между адресом первого элемента строки и именем, как это было в случае строк старого стиля, то есть &s[0] не равно s.

Операции

Кроме операции индексации, для доступа к элементу строки определена функция at:

string s("Вася");
cout « s.at(1); // Будет выведен символ а

Операции

Если индекс превышает длину строки, порождается исключение out_of_range.

Операции

Если индекс превышает длину строки, порождается исключение out_of_range.

Операции

Для работы со строками целиком этих операций достаточно, а для обработки частей строк (например, поиска подстроки, вставки в строку, удаления символов) в классе string определено множество разнообразных методов (функций).

Функции

Функции класса string для удобства рассмотрения можно разбить на несколько категорий: присваивание и добавление частей строк, преобразования строк, поиск подстрок, сравнение и получение характеристик строк.

Присваивание и добавление частей строк

Для присваивания части одной строки другой служит функция assign:

```
assign(const string& str);

assign(const string& str, size_type pos, size_type n);

assign(const char* s, size_type n);
```

Присваивание и добавление частей строк

Первая форма функции присваивает строку str вызывающей строке, при этом действие функции эквивалентно операции присваивания:

```
string s1(“Bacfl”), s2;
s2.assign(s1); // Равносильно s2 = s1;
```

Присваивание и добавление частей строк

Вторая форма присваивает вызывающей строке часть строки str, начиная с позиции pos.

Если pos больше длины строки, порождается исключение out_of_range.

Вызывающей строке присваивается n символов, либо, если pos + n больше, чем длина строки str, все символы до конца строки str.

Присваивание и добавление частей строк

Третья форма присваивает вызывающей строке n символов строки s старого типа.

Присваивание и добавление частей строк

Для добавления части одной строки к другой служит функция append:

```
append(const string& str);

append(const string& str, size_type pos, size_type n);

append(const char* s, size_type n);
```