Строки в языке С

Строка - это последовательность ASCII или UNICODE символов.

Строки в С, как и в большинстве языков программирования высокого уровня рассматриваются как отдельный тип, входящий в систему базовых типов языка. Так как язык С по своему происхождению является языком системного программирования, то **строковый тип данных в С как таковой отсутствует, а в качестве строк в С используются обычные массивы символов**.

Исторически сложилось два представления формата строк:

1. формат ANSI;
2. строки с завершающим нулем (используется в С).

Формат ANSI устанавливает, что значением первой позиции в строке является ее длина, а затем следуют сами символы строки. Например, представление строки "Моя строка!" будет следующим:

```
11  'М'  'о'  'я'  ' '  'с'  'т'  'р'  'о'  'к'  'а'  '!'
```

В строках с завершающим нулем, значащие символы строки указываются с первой позиции, а признаком завершения строки является значение ноль. Представление рассмотренной ранее строки в этом формате имеет вид:

```
'М' 'о' 'я' ' ' 'с' 'т' 'р' 'о' 'к' 'а' '!' 0
```

Строки в языке С++ (класс string)

В языке С++ для удобной работы со строками есть класс string, для использования которого необходимо подключить заголовочный файл string.

Строки можно объявлять и одновременно присваивать им значения:

```
string S1, S2 = "Hello";
```

Строка S1 будет пустой, строка S2 будет состоять из 5 символов.

К отдельным символам строки можно обращаться по индексу, как к элементам массива или С-строк. Например `S[0]` - это первый символ строки.

Для того, чтобы узнать длину строки можно использовать метод size() строки. Например, последний символ строки S это `S[S.size() - 1]`.

Строки в языке С++ могут

Конструкторы строк

Строки можно создавать с использованием следующих конструкторов:
`string()` - конструктор по умолчанию (без параметров) создает пустую строку.
`string(string & S)` - копия строки `S`
`string(size_t n, char c)` - повторение символа `c` заданное число `n` раз.
`string(size_t c)` - строка из одного символа `c`.
`string(string & S, size_t start, size_t len)` - строка, содержащая не более, чем `len` символов данной строки `S`, начиная с символа номер `start`.

Конструкторы можно вызывать явно, например, так:

```
S += string(10, 'z');
```

В этом примере явно вызывается конструктор `string` для создания строки, состоящей из 10 символов `'z'`.

Неявно конструктор вызывается при объявлении строки с указанием дополнительных параметров. Например, так:

```
string S(10, 'z');
```

Подробней о конструкторах для строк читайте здесь.

Ввод-вывод строк

Строка выводится точно так же, как и числовые значения:

```
cout << S;
```

Для считывания строки можно использовать операцию ">>" для объекта cin:

```
cin >> S;
```

В этом случае считывается строка из **непробельных** символов, пропуская пробелы и концы строк. Это удобно для того, чтобы разбивать текст на слова, или чтобы читать данные до конца файла при помощи `while (cin >> S)`.

Можно считывать строки до появления символа конца строки при помощи функции getline. Сам символ конца строки считывается из входного потока, но к строке не добавляется:

```
getline(cin S);
```

Арифметические операторы

Со строками можно выполнять следующие арифметические операции:

- `=` присваивание значения.
- `+=` добавление в конец строки другой строки или символа.
- `+` конкатенация двух строк, конкатенация строки и символа.
- `==`, `!=` посимвольное сравнение.
- `<`, `>`, `<=`, `>=` лексикографическое сравнение.

То есть можно скопировать содержимое одной строки в другую при помощи операции S1 = S2, сравнить две строки на равенство при помощи S1 == S2, сравнить строки в лексикографическом порядке при помощи S1 < S2, или сделать сложение (конкатенацию) двух строк в виде S = S1 + S2.

Подробней об операторах для строк читайте здесь.

Методы строк

У строк есть разные методы, многие из них можно использовать несколькими разными способами (с разным набором параметров).

Рассмотрим эти методы подробней.

size

Метод size() возращает длину длину строки. Возвращаемое значение является беззнаковым типом (как и во всех случаях, когда функция возращает значение, равное длине строке или индексу элемента - эти значения беззнаковые). Поэтому нужно аккуратно выполнять операцию вычитания из значения, которое возвращает size(). Например, ошибочным будет запись цикла, перебирающего все символы строки, кроме последнего, в виде for (int i = 0; i < S.size() - 1; ++i).

Кроме того, у строк есть метод length(), который также возвращает длину строки.

Подробней о методе size.

resize

`S.resize(n)` - Изменяет длину строки, новая длина строки становится равна n. При этом строка может как уменьшится, так и увеличиться. Если вызвать в виде `S.resize(n, c)`, где c - символ, то при увеличении длины строки добавляемые символы будут равны c.

Подробней о методе resize.

clear

`S.clear()` - очищает строчку, строка становится пустой.

Подробней о методе clear.

empty

`S.empty()` - возвращает true, если строка пуста, false - если непуста.

Подробней о методе empty.

push_back

`S.push_back(c)` - добавляет в конец строки символ c, вызывается с одним параметром типа char.

Подробней о методе push_back.

append

Добавляет в конец строки несколько символов, другую строку или фрагмент другой строки. Имеет много способов вызова.

`S.append(n, c)` - добавляет в конец строки n одинаковых символов, равных c. n имеет целочисленный тип, c - char.

`S.append(T)` - добавляет в конец строки S содержимое строки T. T может быть объектом класса string или C-строкой.

`S.append(T, pos, count)` - добавляет в конец строки S символы строки T начиная с символа с индексом pos количеством count.

Подробней о методе append.

erase

`S.erase(pos)` - удаляет из строки S с символа с индексом pos и до конца строки.

`S.erase(pos, count)` - удаляет из строки S с символа с индексом pos количеством count или до конца строки, если pos + count > S.size().

Подробней о методе erase.

insert

Вставляет в середину строки несколько символов, другую строку или фрагмент другой строки. Способы вызова аналогичны способам вызова метода append, только первым параметром является значение i - позиция, в которую вставляются символы. Первый вставленный символ будет иметь индекс i, а все символы, которые ранее имели индекс i и более сдвигаются вправо.

`S.insert(i, n, c)` - вставить n одинаковых символов, равных c. n имеет целочисленный тип, c - char.

`S.insert(i, T)` - вставить содержимое строки T. T может быть объектом класса string или C-строкой.

`S.insert(i, T, pos, count)` - вставить символы строки T начиная с символа с индексом pos количеством count.

Подробней о методе insert.

substr

`S.substr(pos)` - возвращает подстроку данной строки начиная с символа с индексом pos и до конца строки.

`S.substr(pos, count)` - возвращает подстроку данной строки начиная с символа с индексом pos количеством count или до конца строки, если pos + count > S.size().

Подробней о методе substr.

replace

Заменяет фрагмент строки на несколько равных символов, другую строку или фрагмент другой строки. Способы вызова аналогичны способам вызова метода append, только первыми двумя параметрами являются два числа: pos и count. Из данной строки удаляется count символов, начиная с символа pos, и на их место вставляются новые символы.

`S.replace(pos, count, n, c)` - вставить n одинаковых символов, равных c. n имеет целочисленный тип, c - char.

`S.replace(pos, count, T)` - вставить содержимое строки T. T может быть объектом класса string или C-строкой.

`S.replace(pos, count, T, pos2, count2)` - вставить символы строки T начиная с символа с индексом pos количеством count.

Подробней о методе replace.

find

Ищет в данной строке первое вхождение другой строки str. Возвращается номер первого символа, начиная с которого далее идет подстрока, равная строке str. Если эта строка не найдена, то возвращается константа string::npos (которая

равна -1, но при этом является беззнаковой, то есть на самом деле является большим безннаковым положительным числом).

Если задано значение pos, то поиск начинается с позиции pos, то есть возвращаемое значение будет не меньше, чем pos. Если значение pos не указано, то считается, что оно равно 0 - поиск осуществляется с начала строки.

`S.find(str, pos = 0)` - искать первое входение строки str начиная с позиции pos. Если pos не задано - то начиная с начала строки S.

`S.find(str, pos, n)` - искать в данной строке подстроку, равную первым n символам строки str. Значение pos должно быть задано.

Подробней о методе find.

rfind

Ищет последнее вхождение подстроки ("правый" поиск). Способы вызова аналогичны способам вызова метода find.

Подробней о методе rfind.

find_first_of

Ищет в данной строке первое появление любого из символов данной строки str. Возвращается номер этого символа или значение string::npos.

Если задано значение pos, то поиск начинается с позиции pos, то есть возвращаемое значение будет не меньше, чем pos. Если значение pos не указано, то считается, что оно равно 0 - поиск осуществляется с начала строки.

`S.find_first_of(str, pos = 0)` - искать первое входение любого символа строки str начиная с позиции pos. Если pos не задано - то начиная с начала строки S.

Подробней о методе find_first_of.

find_last_of

Ищет в данной строке последнее появление любого из символов данной строки str. Способы вызова и возвращаемое значение аналогичны методу find_first_of.

Подробней о методе find_last_of.

find_first_not_of

Ищет в данной строке первое появление символа, отличного от символов строки str. Способы вызова и возвращаемое значение аналогичны методу find_first_of.

Подробней о методе find_first_not_of.

find_last_not_of

Ищет в данной строке последнее появление символа, отличного от символов строки str. Способы вызова и возвращаемое значение аналогичны методу find_first_of.

Подробней о методе find_last_not_of.

c_str

Возвращает указать на область памяти, в которой хранятся символы строки, возвращает значение типа char*. Возвращаемое значение можно рассматривать как С-строку и использовать в функциях, которые должны получать на вход С-строку.

Подробней о методе c_str.