# String –строка

**Правила инициализации**

При работе со строками им необязательно указывать размер, программа автоматически будет расширятся пока будет вводится текст.

*char first\_date[] = {"Le Chapon Dodu"};*

*char second\_date[] {"The Elegant Plate"};*

*string third\_date = {"The Bread Bowl"}; не работает*

*string fourth\_date {"Hank's Fine Eats"}; не работает*

*String str1 = {“Hello world”};*

*String str2 {“Hello world”};*

Для работы со строками их нужно подключить директиву <string>

**Операция со строками.**

Строки можно присваивать одну к другой.

Добавлять .

Соединять .

**Листинг 4.8. strtype2. срр**

*// strtype2.cpp -- присваивание, сложение, добавление*

*#include <iostream>*

*tinclude <string> // обеспечение доступа к классу string*

*int main()*

*{*

*using namespace std;*

*string si = "penguin";*

*string s2, s3;*

***// Присваивание одного объекта string другому***

*cout « "You can assign one string object to another: s2 = sl\n";*

*s2 = si;*

*cout << "si: " « si << ", s2: " « s2 << endl; \**

***// Присваивание строки в стиле С объекту string***

*cout « "You can assign a C-style string to a string object.\n";*

*cout « "s2 = \"buzzard\"\n";*

*s2 = "buzzard";*

*cout « "s2: " « s2 « endl;*

*/*

***// Конкатенация строк***

*cout << "You can concatenate strings: s3 = si + s2\n";*

*s3 = si + s2;*

*cout « "s3: " << s3 << endl;*

***// Добавление строки***

*cout « "You can append strings. \n";*

*si += s2;*

*cout <<"sl += s2 yields si = " « si « endl;*

*s2 += " for a day";*

*cout <<"s2 += \" for a day\" yields s2 = " « s2 « endl;*

*return 0;*

*}*

Объект string можно инициализировать строкой в стиле С.

* Чтобы сохранить клавиатурный ввод в объекте string, можно использовать сіп.
* Для отображения объекта string можно применять cout.
* Можно использовать нотацию массивов для доступа к индивидуальным символам, хранящимся в объекте string.

**Дополнительные сведения об операциях класса string**

Для того что бы использовать данные функции нужно подключить заголовочный файл cstring (бывший string.h).

*strcpy(charrl, charr2);* ***// копировать charr2 в charrl***

*strcat(charrl, charr2);* ***// добавить содержимое charr2 к charl***

**Дополнительные сведения о вводе-выводе класса string**

**Листинг 4.10. strtype4 .срр**

*// strtype4.cpp -- ввод строки с пробелами*

*#include <iostream>*

*#include <string>*

*#include <cstring>*

*int main ()*

*{*

*using namespace std;*

*char charr[20];*

*string str;*

*// Длина строки в charr перед вводом*

*cout « "Length of string in charr before input: "*

*« strlen(charr) « endl;*

*// Длина строки в str перед вводом*

*cout << "Length of string in str before input: "*

*« str.size () « endl;*

*cout « "Enter a line of text:\n"; // ввод строки текста*

*cin.getline(charr, 20); // указание максимальной длины*

*cout « "You entered: " « charr « endl;*

*cout « "Enter another line of text:\n"; // ввод другой строки текста*

*getline(cin, str); // теперь сіп - аргумент; спецификатор длины отсутствует*

*cout « "You entered: " << str << endl;*

*// Длина строки в charr после ввода*

*cout « "Length of string in charr after input: "*

*« strlen(charr) « endl;*

*// Длина строки в str после ввода*

*cout « "Length of string in str after input: "*

*« str.size () « endl;*

*return 0;*

***}***

Следующий код читает строку в массив:

*cin.getline(charr, 20);*

Точечная нотация указывает на то, что функция get line () — это метод класса istream. (Вспомните, что сіп является объектом класса istream.) Как упоминалось ранее, первый аргумент задает целевой массив, а второй — его размер, используемый get line () для того, чтобы избежать переполнения массива.

Следующий код читает строку в объект string:

*getline (cin,str);*

Здесь точечная нотация не используется, а это говорит о том, что данная функция getline () нг является методом класса. Поэтому она принимает объект сіп как аргумент, сообщающий о том, где искать ввод. К тому же нет аргумента, задающего размер строки, потому что объект string автоматически изменяет свой размер, чтобы вместить строку.

**Другие формы строковых литералов**

*wchar\_t title[] = L"Chief Astrogator"; // строка w\_char*

*charl6\_t name[] = u"Felonia Ripova"; // строка char\_16*

*char32\_t car[] = U"Humber Super Snipe"; // строка char\_32*

С++11 также поддерживает схему кодирования для символов Unicode под названием UTF-8. В зависимости от числового значения, в этой схеме символ может храниться в пределах от 8-битной единице, или октете, до четырех 8-бит-х единиц. Для указания строковых литералов такого типа в C++ используется префикс и8.

Другим добавлением C++11 является **необработанная (raw) строка**. В такой строке символы представляют сами себя. Например, последовательность \n не интерпретируется как представление символа новой строки; вместо этого она будет выглядеть как два обычных символа, обратная косая черта и п, отображаясь таким же образом на экране. Или еще один пример: внутри необработанной строки можно использовать просто двойную кавычку ", а не конструкцию. Естественно, поскольку двойные кавычки можно применять внутри строкового

литерала, их больше нельзя использовать для обозначения начала и конца строки. Таким

образом, в необработанных строках в качестве разделителей применяются последовательности " ( и ) ", а также префикс R для идентификации их как необработанных строк:

*cout << R" (Jim "King" Tutt uses "\n" instead of endl.) " << '\n';*

**Этот оператор приводит к отображению следующей строки:**

*Jim "King" Tutt uses \n instead of endl.*

Эквивалентный стандартный строковый литерал выглядит так:

*cout << "Jim \"King\" Tutt uses \" \\n\" instead of endl." << '\n';*

*cout << R"+\*("(Who wouldn't?)", she whispered.)+\*" << endl;*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Листинг 4,9. strtype3. срр**

*// strtype3.cpp — дополнительные средства класса string*

*#include <iostream>*

*#include <string> // обеспечение доступа к классу string*

*#include <cstring> // библиотека обработки строк в стиле С*

*int main ()*

*{*

*using namespace std;*

*char charrl[20];*

*char charr2[20] = "jaguar";*

*string strl;*

*string str2 = "panther";*

*Составные типы 149*

*150 Глава 4*

*// Присваивание объектов string и символьных массивов*

*strl = str2; // копирование str2 в str2*

*strcpy(charrl, charr2); // копирование charr2 в charrl*

*// Добавление объектов string и символьных массивов*

*strl += " paste"; // добавление " paste" в конец strl*

*strcat(charrl, " juice"); // добавление " juice" в конец charrl*

*// Определение длины объекта string и строки в стиле С*

*int lenl = strl.sizeO; // получение длины strl*

*int Ien2 = strlen(charrl); // получение длины charrl*

*cout « "The string " « strl « " contains "*

*« lenl « " characters. \n";*

*cout << "The string " << charrl << " contains "*

« len2 « " characters. \n";

*return 0;*

*}*

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

Получения длины строки

Функция *strlen () —* это стандартная функция, которая принимает в качестве аргумента

строку в стиле С и возвращает количество символов в ней.

Функция *size ()* обычно делает то же самое, но синтаксис ее вызова отличается.

*int lenl = strl.size();* ***// получение длины strl***

*int Ien2 = strlen(charrl);* ***// получение длины charrl***