СТРОКИ КЛАССА STRING

Строковый тип в C/C++ – **отсутствует**!

Моделирование строк:

① `char СТРОКА [длина];  // С-строка, строка в стиле С,`
`// массив символов с нулевым байтом в конце`

② объект класса `string` (ISO /ANSI C++98, для работы с 8-битовыми строками)

Почему **не хватает** строк в стиле С?

① К строкам в стиле С не применимы стандартные операторы языка C++
(=> С-строки не могут быть частью выражений):

```
char s1[80], s2[80], s3 [80]; // '\0'
s1 = "Green"; // Нельзя! НО: char s1[80] {"Green"}
s2 = "Yellow";  // Нельзя!
s3 = s1 + s2;   // Ошибка, и так нельзя!
```

Однако:

```
strcat(s3, s2);                    strcpy(s2, "Yellow");
strcpy(s1, "Green");               strcpy(s3, s1);
```

② Возможны ошибки работы с памятью

```
strcpy(s3, s1);
```

Причины, **включения** класса `string` в язык C++:

- логичность (теперь строка является типом данных),
- удобство (к строкам можно применять стандартные операторы языка C++),
- безопасность (невозможно выйти за пределы массива).

Пример. Сходства и различия между объектами `string` и символьными массивами.

```
#include <iostream>
#include <string>
int main() {
    using namespace std;
    char c_str1[20]; //создание пустого массива
    char c_str2[20] = "jaguar";
    string str1;
    string str2 = "puma";
    cout << "Введите животное из семейства кошачьих: ";
    cin >> c_str1;
    cout << "Введите другое животное из семейства кошачьих: ";
    cin >> str1;
    cout << "Кошки:\n";
    cout << c_str1 << " " << c_str2 << " "
         << str1 << " " << str2
         << endl;
    cout << "Третий символ в строке " << c_str2 << ": "
         << c_str2[2] << endl;
    cout << "Третий символ в строке " << str2 << ": "
         << str2[2] << endl;
    return 0;
}
```

Объявление в виде переменной, а не массива

инициализация С-строкой

использование cin для ввода

использование cout для вывода

нотация массивов для доступа к символам

str1 увеличен для того, чтобы вместить ввод; отсутствие явного указания размера

обеспечение доступа к классу string

Результаты работы программы

```
Введите животное из семейства кошачьих: leopard
Введите другое животное из семейства кошачьих: tiger
Кошки:
leopard jaguar tiger puma
Третий символ в строке jaguar: g
Третий символ в строке puma: m
```

Выводы из примера:

- Сходства: объект `string` можно использовать так же, как символьный массив.
- Отличия: объект `string` - обычная переменная, а не массив.
- Особенности: заголовочный файл `string`.
  `string` – часть пространства имен `std` (директива `using` или `std::string`)
  отсутствие явного указания размера строки `string`

Инициализация строк в C++11

```
#include <iostream>
#include <string>
int main() {
  using namespace std;
  string one("Proverb.");
  cout << one << endl;
  string two(20, '*');
  cout << two << endl;
string three {"trouble "};
string four(three);
three = "Don`t " + four + four;
three += "\nuntil " ;
cout << three;
  string five;
  five = four;
  five[0] = 'T';
  cout << five;
string six(&five[1], &five[7]);
six = "t" + six + 's';
cout << six;
char you[] = " you!";
string seven(you, 10);
cout << seven << "\n";
  string ten(two, 3, 14);
  cout << ten << " End *" << endl;
  return 0;
}
```

`string (const char * s)`
Инициализация С-строкой, которая указана в `s`.

`string (size_type n, char c)`
Создает объект типа `string` из `n` символов `c`.

Списковая инициализация

`string(const string & str)`
Инициализация объектом `str` типа `string` (конструктор копирования)

Использование `string` в выражениях: = и +=

`string()` Создает объект типа `string` нулевого размера (конструктор по умолчанию)

Нотация массива доступа к символам

Инициализация символами `five` в диапазоне `[1, 7)`

`string (const char * s, size_type n)`
Инициализация С-строкой в `s` и содержит `n` символов, даже если `n` превышает длину `s`.

Инициализация объектом `two`, начиная с позиции `3` и оканчивая концом `two`, либо ограничиваясь `14` символами, в зависимости от того, какое условие будет удовлетворено раньше

Результаты работы программы

```
Proverb.
********************
Don`t trouble trouble
until Trouble troubles you!    ◄ ☺►
************** End *
```

Ввод данных

Для **С-строк**:

```
char info[100];
cin >> info; // чтение слова до пробельного символа.
             // Проблемы, если вводим > 100 символов
cin.getline(info, 100); // чтение строки с отбрасыванием символа \n
cin.get(info, 100);     // чтение строки с сохранением символа \n в очереди
cin.getline(info, 100, ':'); // чтение до символа
```

Для объектов **`string`**:

```
string stuff;
cin >> stuff;            // чтение слова с сохранением символа \n в очереди
getline(cin, stuff);     // чтение строки с отбрасыванием символа \n
getline(cin, stuff, ':'); // чтение до символа
```

Пример.

Что делает программа?

```
#include <iostream>
#include <fstream>
#include <string>
#include <cstdlib>
int main() {
   using namespace std;
   ifstream fin;
   fin.open("tobuy.txt");
   if (fin.is_open() == false) {
      cout << "Can't open file. Bye.\n";
      exit(EXIT_FAILURE);
   }
   string item;
   int count = 0;
   getline(fin, item);
   while (fin) {
      ++count;
      cout << count <<" : " << item << endl;
      getline(fin, item);
   }
   cout << "Done\n";
   fin.close ();
   return 0;
}
```

Что же еще можно делать со строками string?

Строки можно сравнивать:

==	Равенство
!=	Неравенство
<	Меньше
<=	Меньш е или равно
>	Больше
>=	Больше или равно

```
string snake1("cobra");
string snake2("coral");
char snake3[20] = "anaconda";
if (snake1 < snake 2) // string < string
if (snake1 == snake3) // string == C-строка
if (snake3 != snake2) // C-строка != string
```

Длина строки:

```
if (snake1.length() == snake2.size())
cout << " Строки имеют одинаковую длину.\n";
```

Поиск подстроки или символа в строке:

!!! `string::npos` — максимально возможное количество символов в строке.
Как правило, это наибольшее значение типа `unsigned int` или `unsigned long`.
18446744073709551615

```
string snake1("cobra");
```

`find_first_of()` отыскивает первое вхождение в строке любого из символов, переданных в аргументах метода:

```
int where = snakel.find_first_of("hark");
// 3 – позиция символа r в строке "cobra"
```

`find_last_of()` – аналогично, только находит последнее вхождение:

```
int where = snakel.last_first_of("hark");
// 4 - позиция символа a в строке "cobra"
```

`find_first_not_of()` находит первый символ в вызывающей строке, который отличается от символа, переданного в аргументе. Таким образом, приведенный ниже оператор вернет:

```
int where = snakel.find_first_not_of("hark");
// 0 - позиция символа c в cobra, поскольку символ c не найден в hark
```

`find_last_not_of()` – аналогично, только находит последнее вхождение:

```
int where = snakel.find_last_not_of("hark");
// 2 - позиция символа b в cobra, поскольку символ c не найден в hark
```

Другие методы для работы со строками

+= или `append()`	Методы присоединения и добавления
`assign()`	Дополнительный метод присваивания
`insert()`	Методы вставки
`erase()`	Методы очистки
`replace()`	Методы замены
`copy()`	Метод копирует объект `string` или его часть в целевой массив символов
`swap()`	Метод производит обмен содержимого двух объектов `string` с помощью алгоритма с константным временем

http://www.cplusplus.com/reference/algorithm/swap/
https://en.cppreference.com/w/cpp/algorithm/swap

Игра «Палач»

```
// hangman.epp – использование некоторых методов работы со строками
...
using std::string;
const int NUM = 13;
const string wordlist[NUM] = {"danger", "florid", "insult",
      "loaner", "onset", "plaid", "quilt", "stolid", "train", "useful",
      "valid", "yearn", "tiger"};
int main() {
   using std::cout;   using std::cin;   using std::endl;
   std::srand(std::time(0));
      string target = wordlist[std::rand() % NUM];
      int length = target.length();
      string attempt(length, '-');
      string badchars;
      int guesses = 6;
```

```
        cout << "Guess my secret word. It has " << length
           << " letters, and you guess\n"
           << "one letter at a time. You get " << guesses
           << " wrong guesses.\n";
        cout << "Your word: " << attempt << endl; // вывод слова
        while (guesses > 0 && attempt != target) {
           char letter;
           cout << "Guess a letter: ";
           cin >> letter;
           if (badchars.find(letter) != string::npos) {
              cout << "You already guessed that. Try again.\n" ;
              continue;
           }
           int loc = target.find(letter);
           if (loc == string::npos) {
              cout << "Oh, bad guess !\n";
              --guesses;
              badchars += letter;
           } else {
              cout << "Good guess!\n";
              attempt[loc]=letter;
           }
           cout << "Your word: " << attempt << endl;
           if (attempt != target) {
              if (badchars.length() > 0)
                 cout << "Bad choices: " << badchars << endl;
              cout << guesses << "bad guesses left\n" ;
           }
        }
        if (guesses > 0)
           cout << "That's right!\n";
        else
           cout << "Sorry, the word is " << target << " \n" ;
     cout << "Bye\n";
     return 0;
}
```

работа со строками так же, как с числовыми переменными

find():
- для проверки на повторное использование символа
- проверка на наличие введённого символа в загаданном слове

npos – для индикации неудачного поиска символа в строке

Нотация массивов для сохранения буквы в соответствующую позицию строки ответа.

Дружественный интерфейс: вывод не верно введённых символов, если они есть.

Результаты работы программы:

```
Guess my secret word. It has 6 letters, and you guess
one letter at a time. You get 6 wrong guesses.
```

```
Your word: ------
Guess a letter: a
Good guess!
Your word: -a----
6bad guesses left
Guess a letter: w
Oh, bad guess !
Your word: -a----
Bad choices: w
5bad guesses left
Guess a letter: g
Good guess!
Your word: -a-g--
Bad choices: w
5bad guesses left
Guess a letter: p
Oh, bad guess !
Your word: -a-g--
Bad choices: wp
4bad guesses left
```

```
Guess a letter: d
Good guess!
Your word: da-g--
Bad choices: wp
4bad guesses left
Guess a letter: n
Good guess!
Your word: dang--
Bad choices: wp
4bad guesses left
Guess a letter: e
Good guess!
Your word: dange-
Bad choices: wp
4bad guesses left
Guess a letter: r
Good guess!
Your word: danger
That's right!
```

```
Bye
```

Упражнения по программированию.

Измените программу так, чтобы:

1. Исходные слова хранились в текстовом файле. Считывание происходило бы только одного «загаданного» слова. Структура файла – любая.
2. Можно было «загадывать» слова с повторяющимися буквами. Например, "elephant". При вводе буквы 'e' открывались все позиции вхождения этой буквы в слово.
3. Информация о ранее введённых буквах была бы верной.
4. Программа давала возможность сыграть несколько раз.

Срок – **до 09.06**.

Для проверки работы программы необходим исходный код и файл со словами.