Строки

В **C++** нет стандартного типа данных «строка», но поддерживает массив символов с “/0” в конце. Функции в библиотеке <string.h> (<cstring)

Тип данных **string:**

- Строки типа **string** защищены от выхода информации за их границы.
- С ними можно работать так же, как с любым встроенным типом данных.
- Для использования класса необходимо подключить к программе заголовочный файл <**string**>.

Пример использования С и С++

```
#include<cstring>
#include<string>
#1nclude<iostream>
using namespace std;
int main() {
   charcl[80],c2[80],c3[80];//Строки с завершающим нулем
   string s1,s2,s3;
   strcpy(cl,"old string one"); //Присваивание строк
   strcpy(c2,cl);
   s1="new strin gone";
   s2=sl;
   strcpy(c3,cl) ; //Конкатенация строк
   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;
}
```

Конструкторы и присваивание строк

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

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

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

Операции

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

- Для строк типа **string** не соблюдается соответствие между адресом первого элемента строки и именем.
- Кроме операции индексации, для доступа к элементу строки определена функция at:

```
string s("Вася");
cout << s.at(1);
```

out_of_range

Функции

Присваивание части одной строки другой:

```
assign(const string& str);
assign(const string& str, size_type pos, size_type n);
assign(const char* s, size_type n);
```

Добавление части одной строки к другой:

```
append(const string& str);
append(const string& str, size_type pos, size_type n);
append(const char* s, size_type n);
```

самое большое положительное число типа size_type

Вставка в одну строку части другой строки:

insert(size_type pos1, const string& str);

insert(size_type pos1,const string& str,size_type pos2, size_type n);

insert(size_type pos, const char* s, size_type n);

Удаление части строки:

erase(size_type pos = 0, size_type n = npos);

Обмен содержимого двух строк:

swap(string& s);

самое большое положительное число типа size_type

Выделение части строки :

string substr(size_type pos = 0,size_type n = npos) const;

Преобразование string в строки старого стиля:

const char* c_str() const;

const char* data() const;

(data не добавляет в конец строки нуль-символ)

Функция **copy** копирует в массив **s n** элементов вызывающей строки, начиная с позиции **pos**. Нуль-символ в результирующий массив не заносится:

size_type copy(char* s, size_type n, size_type pos = 0) const;

Пример

```
#include <string>
#include <iostream>
using namespace std;
int main (){
   string s1("прекрасная королева"), s2("ле"),
  s3("корова");
   cout << "s3= " << s3.insert(4, s2) << endl;
   cout << "s3= " << s3.insert(7, "к") << endl;
   s1.erase(0,3);
   cout << "s1= " << s1.erase(12,2) << endl;
   cout << "s1= " << s1.replace(0,3, s3, 4,2) <<
  endl;
}
```

```
s3= королева
s3= королевка
s1= красная корова
s1= лесная корова
```

Поиск подстрок

1. size_type find(const string& str,size_type pos = 0) const;
2. size_type find(char c, size_type pos = 0) const;
3. size_type rfind(const string& str,size_type pos=npos) const;
4. size_type rfind(char c, size_type pos = npos) const;
5. size_type find_first_of(const string& str,size_type pos=0) const;
6. size_type find_first_of(char c, size_type pos = 0) const;
7. size_type find_last_of(const string& str, size_type pos = npos) const;
8. size_type find_last_of(char c, size_type pos = npos) const;
9. size_type find_first_not_of(const string& str, size_type pos = 0) const;
10. size_type find_first_not_of(char c, size_type pos = 0) const;
11. size_type find_last_not_of(const string& str, size_type pos = npos) const;
12. size_type find_last_not_of(char c,size_type pos=npos) const;

Пример

```
char GetColumn() {
    string goodChar = "ABCDEFGH";
    char symb;
    cout << "Введите обозначение колонки: ";
    while (1) {
        cin >> symb;
        if (-1 == goodChar.find(symb)) {
            cout << "Ошибка. Введите корректный
  символ:\n";
            continue;
        }       return symb;
    }}
```

Сравнение частей строк

Для сравнения строк целиком применяются перегруженные операции отношения, а если требуется сравнивать части строк, используется функция **compare**:

```
int compare(const string& str) const;
int compare(size_type pos1, size_type n1,
     const string& str) const;
int compare(size_type pos1, size_type n1,
const string& str, size_type pos2, size_type
n2) const;
```

Аналогичные формы функций существуют и для сравнения строк типа **string** со строками старого стиля.

Получение характеристик строк

```
// Размер строки
    size_type size() const;
// Количество элементов строки
    size_type length() const;
// Максимальная длина строки
    size_type max_size() const;
// Объем памяти, занимаемый строкой
    size_type capacity() const;
// Истина, если строка пустая
    bool empty() const;
```