Программирование

Тема 7.1 Строки в C++

Строки string в C++

Пример 1: Способы создания строк, операции со строками, ввод – вывод строк

```
строка1 строка2
строка1 строка2
строка1

Введите строку str1: string 1
Введите строку str2: string 1
Строки равны
```

```
#include <iostream>
#include <string>
using namespace std;

int main()
{ char s[] = "строка1 ";
  // cout << "Введите строку s: "; cin.getline(s,10);
  string s1(s); //объект типа string инициализируется си-строкой
  string s2("строка2 ");
  string s3(s1); //создает объект s3 на основе объекта s1
  s3+=s2;        cout << s3 << endl;
  s3[0] = 's';   cout << s3 << endl;;
  s2 = s1;       cout << s2 << endl;
  string str1, str2;//создаются пустые объекты типа string
  cout << "\nВведите строку str1: ";
  getline(cin, str1);
  cout << "Введите строку str2: ";
  getline(cin, str2);
  if (str1 == str2) cout << "Строки равны " << endl;
  if (str1 != str2) cout << "Строки не равны: ";
  if (str1 > str2) cout << "строка str1 больше строки str2" << endl;
  if (str1 < str2) cout <<"строка str1 меньше строки str2" << endl;
  return 0;
}
```

Строки string в C++

- Кроме работы со строками, как с массивом символов, в C++ существует специальный класс string.
- Для работы с ним необходимо в начале программы подключить заголовочный файл string и пространство имен std:

```
#include <string>
using namespace std;
```

- Конструкторы класса string позволяют создавать пустую строку, инициализировать строку си-строкой или ранее созданным объектом типа string (пример 1).
- Ввод и вывод строк типа string осуществляется с помощью потоковых операций >> и <<. Но операция ввода (>>) позволяет вводить строку только до первого пробела, поэтому для ввода строк типа string удобно использовать функцию getline() (пример 1).
- Для строк типа string определены следующие операции:
  - присваивания (= , +=);
  - сравнения (== , != , < , >);
  - обращение по индексу ([ ]).
- Операция простого присваивания (=) позволяет в одну строку записать содержимое другой строки.
- Операция составного присваивания (+=) позволяет в конец одной строки добавить содержимое другой строки.
- Операции сравнения (== , != , < , >) выполняют сравнение строк в лексикографическом порядке (по алфавиту). Результатом выполнения операций сравнения являются значение 0, если строки равны; значение < 0, если строка лексикографически меньше со сравниваемой строкой; значение >0, если строка лексикографически больше со сравниваемой строкой (пример 1).
  - Операция обращения по индексу ([ ]) выполняет доступ с символу строки как для его чтения так и для его изменения (пример 1).

Пример 2: Изменение строк

```
Розовые лотосы удивительно красивы
Розовые лотосы, кажется, можно есть
```

```
#include <iostream>
#include <string>
using namespace std;

int main()
{ string s1 = "Розовые лотосы удивительно красивы";
  string s2 = "можно ";
  string s3 = "есть ";
  string s4 = ", кажется,";
  cout << s1 << endl;
  s1.erase(15, 12); // удаляем слово "удивительно"
  s1.replace(15, 7, s2); // заменяем слово "красивы" на
                         // "можно"
  s1.append(s3);     // добавляем слово "есть"
  s1.insert(14, s4); //добавляем слово ", кажется,"
                     // после слов "Розовые лотосы"
  cout << s1 << endl;
  return 0;
}
```

Пример 3: Поиск в строке

```
Позиция вхождения символа a: 0
Позиция вхождения строки klmnop: 10
Строка klmnop не найдена!
```

```
#include <iostream>
#include <string>
using namespace std;

int main()
{ string ABC = "abcdefghijklmnopqrstuvwxyz";
  char ch = 'a'; char str1[] = "klmnop";
  string str2 = "klmnop";
  // поиск указанного элемента (символа, си-строки,с++-строки)
  if(ABC.find(ch) != string::npos)
      cout << "\nПозиция вхождения символа " << ch << ": " << ABC.find(ch);
     else cout << "\nСимвол " << ch << " не найден!";
  if(ABC.find(str1) != string::npos)
      cout << "\nПозиция вхождения строки " << str1 << ": " << ABC.find(str1);
     else cout << "\nСтрока " << str1 << " не найдена!";
  if(ABC.find(str2,15) != string::npos)
     cout << "\nПозиция вхождения строки " << str2 << ": " << ABC.find(str2,15);
    else cout << "\nСтрока " << str2 << " не найдена!";
```

Пример 3: Поиск в строке

```
  // Поиск символа не входящего в указанную строку
  if(ABC.find_first_not_of("eiou") != string::npos)
     cout << "\nПозиция вхождения символа " <<  ABC.find_first_not_of("eiou");
    else cout << "\nСимволы  не найдены!\n";
  if(ABC.find_last_not_of("eiou") != string::npos)
     cout << "\nПозиция вхождения символа " <<  ABC.find_last_not_of("eiou");
    else cout << "\nСимволы  не найдены!\n";

  // Поиск символа входящего в указанную строку
  if(ABC.find_first_of("eiou") != string::npos)
     cout << "\nПозиция вхождения символа " <<  ABC.find_first_of("eiou");
    else cout << "\nСимволы  не найдены!\n";
  if(ABC.find_last_of("eiou") != string::npos)
     cout << "\nПозиция вхождения символа " <<  ABC.find_last_of("eiou");
    else cout << "\nСимволы  не найдены!\n";
  // функции, которые принимают диапазон в качестве аргументов
  // используют npos в качестве индикатора "все до конца строки"
   string ABC2(ABC, 3, std::string::npos);
          std::cout << '\n'<< ABC2 ;
  return 0;
}
```

```
Позиция вхождения символа 0
Позиция вхождения символа 25
Позиция вхождения символа 4
Позиция вхождения символа 20
defghijklmnopqrstuvwxyz
```

Пример 4: Сравнение строк методом compare()

```
#include <iostream>
#include <string>
using namespace std;

int main()
{ string str1, str2;
    int res, pos = 0, n = 3;
    cout << "Введите строку str1:"; cin >> str1;
    cout << "Введите строку str2:"; cin >> str2;
    // сравнение строк str2 и str1
    res = str1.compare(str2);
    if (res == 0) cout << "Строки равны " << endl;
        else cout << "Строки не равны" << endl;
    // сравнение строки str2 и n символов строки str1,
    // начиная с символа str1[pos]
    res = str1.compare(pos, n, str2);
    if (res == 0) cout << "Строки равны " << endl;
      else cout << "Строки не равны" << endl;
  return 0;
}
```

```
Введите строку str1:string
Введите строку str2:str
Строки не равны
Строки равны
```

Строки string в C++

- Изменение строк типа string выполняется с помощью методов:
  - методы append - добавляют подстроку в конец строки;
  - методы erase - удаляют фрагмент из строки;
  - методы insert - вставляют подстроку в указанное место строки;
  - методы replace - заменяют одну подстроку на другую.
- В примере 2 показано использование методов изменения строк типа string.

- Пример 3 демонстрирует использование методов поиска в строке.
- Методы find выполняют поиск в исходной строке указанного элемента (символа, си-строки, с++-строки). Методы возвращают позицию найденного элемента или npos.
- string::npos - это специальное значение, равное максимальному значению, которое может предоставить тип size_type. Точный смысл данного значения зависит от контекста, но, как правило, оно используется либо как индикатор конца строки в функциях, которые ожидают позицию символа, либо как индикатор ошибки в функциях, которые возвращают позицию в строке.
- Также есть аналогичные find методы rfind, делающие поиск справа налево.
- Методы find_first_not_of и методы find_last_not_of выполняют поиск символа не входящего в указанную строку. Методы возвращают позицию найденного символа или npos.
- Методы find_first_of и методы find_last_of выполняют поиск символа входящего в указанную строку. Методы возвращают позицию найденного символа или npos.
- Для сравнения строк можно использовать мето д compare (пример 4). Метод compare так же как и операции сравнения сравнивает строки в лексикографическом порядке.
- Результатом выполнения метода compare являются значение 0, если строки равны; значение < 0, если строка лексикографически меньше со сравниваемой строкой; значение >0, если строка лексикографически больше со сравниваемой строкой.

Пример 5: Использование методов length(), capacity() и empty()

```
#include <iostream>
#include <string>
#include <fstream>
using namespace std;

int main()
{    bool b = false;
     ifstream fp("E:\\Dir\\info.txt");
     if(! fp.is_open()) return 1;
     string s;
     while(getline(fp, s, '\n') != 0)
     {     cout << "Длина строки - ";
           cout << s.length() << "; макс. длина - ";
           cout << s.capacity() << endl;
     }
     fp.close();
     b = s.empty();
     if (b == true) cout << "\nСтрока пустая!!!";
      else  cout << s << endl;
  return 0;
}
```

info.txt

```
1   Однажды в студеную зимнюю пору
2   Я из лесу вышел - был сильный мороз.
```

```
Длина строки - 30; макс. длина - 30
Длина строки - 36; макс. длина - 60

Строка пустая!!!
```

Пример 6: Массив строк. Использование метода at()

```
#include <iostream>
#include <string>
#include <fstream>
using namespace std;

int main()
{    int k = 0;
     ifstream fp("E:\\Dir\\info.txt");
     if(! fp.is_open()) return 1;
     string mas_str[5];
     while(fp!= 0)
       getline(fp, mas_str[k++], '\n');
     fp.close();
     for(int i=0;i<k;i++)
        cout <<  mas_str[i] << endl;
     try { mas_str[1].at(35)='!';
             cout << mas_str[1];
             }
      catch(...)
        {cout << "\nОшибка: std::out_of_range";}
  return 0;
}
```

```
Однажды в студеную зимнюю пору
Я из лесу вышел - был сильный мороз.

Я из лесу вышел - был сильный мороз!
```

Пример 7: Преобразование строки string в си-строку и обратно

```
#include <iostream>
#include <string>
#include <cstring>
#include <sstream>
using namespace std;
int main()
{   string str = "Я из лесу вышел, - был сильный мороз.";
    char * si_str = new char[str.size()+1];
    strcpy(si_str, str.c_str() );
    char * tokenPtr;
    cout << "Слова: " << endl;
    tokenPtr = strtok(si_str, " ;-,.");
    stringstream s;
    while (tokenPtr != NULL)
     { cout << tokenPtr << endl;
       s << tokenPtr << ' ';
       tokenPtr = strtok(NULL," ;-,.");
     }
    string str2 = s.str();
    cout << "str2: " << str2;
  return 0;
}
```

```
Слова:
Я
из
лесу
вышел
был
сильный
мороз
str2: Я из лесу вышел был сильный мороз
```

Контрольные вопросы

1. Строки string в C++: понятие, способы создания, допустимые операции, основные методы класса string. Примеры.