Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра программной инженерии

 Реферат

По дисциплине «Основы алгоритмизации и программирования»

На тему «Функции, работающие со строками»

 Выполнила:

Студентка1 курса 6 группы

Альшевская Алина Михайловна

Преподаватель: доц. Белодед Н.И

2023, Минск

Функции, работающие со строками

**Функция strlen()**

Функция strlen() является одной из наиболее часто используемых функций для работы со строками в С++. Она принимает один аргумент - указатель на строку - и возвращает длину этой строки. Длина строки определяется как количество символов до первого символа null (‘\0’), который обычно обозначает конец строки.

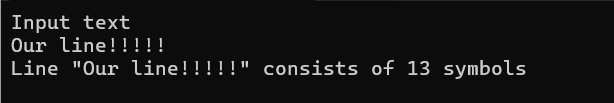
Пример кода без использования функции strlen() для подсчёта длины строки

|  |
| --- |
| #include <iostream>  using namespace std;  int main()  {  char str[128] = "";  cout << "Input text\n";  cin.getline(str, 128);  int symbolCounter = 0;  while (str[symbolCounter] != '\0')  {  symbolCounter++;  }  cout << "Line \"" << str << "\" consists of " << symbolCounter << " \n";  return 0;  } |

Пример кода для подсчёта длины строки с использованием функции strlen()

|  |
| --- |
| #include <iostream>  #include <cstring>  using namespace std;  int main()  {  char str[128] = "";  cout << "Input text\n";  cin.getline(str, 128);  cout << "Line \"" << str << "\" consists of " << strlen(str)<< " symbols\n";  return 0;  } |

Результат первой и второй программы будет одинаковый



**Функция strcat()**

Функция strcat() используется для конкатенации, или объединения, двух строк. Она принимает два аргумента: первый - это строка, к которой будет добавлена вторая строка, а второй - это строка, которую нужно добавить.

Пример кода без использования функции strcat() для объединения двух строк

|  |
| --- |
| #include <iostream>  using namespace std;  int main()  {  char str1[50] = "Hello";  char str2[] = ", World!";    int symbolCounter1 = 0;  while (str1[symbolCounter1] != '\0')// ищем конец первой строки  {  symbolCounter1++;  }  int symbolCounter2 = 0;  while (str1[symbolCounter2] != '\0')// после конца первой строки дописываем вторую  {  str1[symbolCounter1] = str2[symbolCounter2];  symbolCounter1++;  symbolCounter2++;  }  cout << "Concatenated line \"" << str1<< "\"\n";  return 0;  } |

Пример кода с использования функции strcat() для объединения двух строк

|  |
| --- |
| #include <iostream>  #include <cstring>  using namespace std;  int main()  {  char str1[50] = "Hello";  char str2[] = ", World!";    cout << "Concatenated line \"" << strcat(str1,str2)<< "\"\n";  return 0;  } |

Результат первой и второй программы будет одинаковый

****

**Функция strcmp()**

Функция strcmp() сравнивает две строки и возвращает целое число, которое указывает на относительный порядок строк. Если возвращаемое значение меньше нуля, первая строка меньше второй. Число -1 возвращается и  тогда, когда длина строк равна, но символы строк не совпадают. Если оно равно нулю, строки равны. Если оно больше нуля, первая строка больше второй.

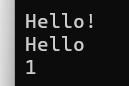
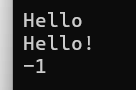
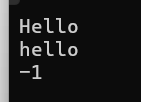
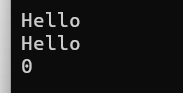
Пример кода без использования функции strcmp() для сравнения двух строк

|  |
| --- |
| #include <iostream>  using namespace std;  int main()  {  char str1[] = "Hello!";  char str2[] = "Hello";  cout << str1 << "\n" << str2 << endl;  int compare = 0, symbolCounter = 0;  while (true)  {  if (strlen(str1) < strlen(str2))  {  cout << --compare << endl;  break;  }  else if (strlen(str1) > strlen(str2))  {  cout << ++compare << endl;  break;  }  else//проверяем равны ли строки если имеют равное количество символов  {  if (str1[symbolCounter] == str2[symbolCounter])// сравниваем посимвольно  {  symbolCounter++;  if (str1[symbolCounter] == '\0' && str2[symbolCounter] == '\0')  {  cout << compare << endl;  break;  }  }  else// если гдето встретились различия  {  cout << --compare << endl;  break;  }  }  }  return 0;  } |

Пример кода с использования функции strcmp() для сравнения двух строк

|  |
| --- |
| #include <iostream>  #include <cstring>  using namespace std;  int main()  {  char str1[] = "Hello!";  char str2[] = "Hello";  cout << str1 << "\n" << str2 << endl;  cout << strcmp(str1, str2) << endl;  return 0;  } |

Результаты первой и второй программы будет одинаковый



**Функция strcpy()**

Функция strcpy() используется для копирования одной строки в другую. Она принимает два аргумента: первый - это строка, в которую будет скопирована вторая строка, а второй - это строка, которую нужно скопировать.

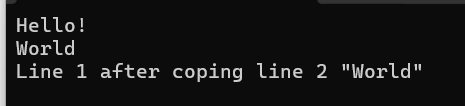
Пример кода без использования функции strcpy() для копирование одной строки в другую

|  |
| --- |
| #include <iostream>  using namespace std;  int main()  {    char str1[50] = "Hello!";  char str2[] = "World";  cout << str1 << "\n" << str2 << endl;  int symbolCounter = 0;  while (true) {  str1[symbolCounter] = str2[symbolCounter];  if (str2[symbolCounter] == '\0') break;  symbolCounter++;  }  cout <<"Line 1 after coping line 2 \""<< str1<<"\"\n" << endl;  return 0; |

Пример кода с использования функции strcpy() для копирование одной строки в другую

|  |
| --- |
| #include <iostream>  #include <cstring>  using namespace std;  int main()  {    char str1[50] = "Hello!";  char str2[] = "World";  cout << str1 << "\n" << str2 << endl;    cout << "Line 1 after coping line 2 \"" << strcpy(str1,str2) << "\"\n" << endl;  return 0;  } |

Результат первой и второй программы будет одинаковый



Функции для работы со строками в C++ имеют ряд преимуществ:

1. **Простота использования**: Эти функции предоставляют простой и удобный способ выполнения основных операций со строками, таких как определение длины строки, конкатенация двух строк, сравнение двух строк и копирование одной строки в другую.
2. **Эффективность**: Функции для работы со строками в C++ обычно оптимизированы для быстрого выполнения, что делает их эффективным выбором для манипуляций со строками.
3. **Стандартизация**: Поскольку эти функции являются частью стандартной библиотеки C++, они доступны в любой среде, поддерживающей C++. Это обеспечивает стандартизацию и совместимость кода.
4. **Универсальность**: Функции для работы со строками могут работать с любыми строками C-стиля, что делает их универсальными инструментами для работы со строками в C++.

Однако стоит отметить, что эти функции не выполняют проверку границ, поэтому при использовании их необходимо убедиться, что наши строки достаточно велики, чтобы вместить результаты этих функций. В противном случае это может привести к ошибкам времени выполнения.