Практическое задание 12
Работа с текстовыми данными в C++.
Объектный тип std::string: практическое использование

Цель. Изучение способов представления символов и строк текста. Знакомство с объектным типом std::string. Практическое использование методов и операций string для реализации алгоритмов работы с текстом. Разработка функций для решения практических задач обработки текста.

ПЛАН

1. Краткое теоретическое введение 1
2. Объектный тип `std::string` 1
3. Алгоритмы работы с текстом 3
4. Функции работы с текстом 3
5. Работа с текстовыми файлами 3

1. Краткое теоретическое введение

В C++ текстовые данные представлены как:
1) символьные константы;
2) символьные переменные;
3) строки – константы;
4) строковые переменные (объекты библиотеки `std`).

Допустимы строковые переменные как массивы символов `*char.`

Символьные константы и переменные, это данные типа `char` – 1-байтовые данные целого типа (2 байта в Юникод).

Символьная константа, это лексема – состоит из изображения символа в одинарных кавычках, например `'': '*'  '?'  'f'  'Я'  '~'  '#'  '1'.`

Объявление символьной переменной:

```
char
unsigned char
```

Внутреннее представление данных `char` – целочисленный код символа. Согласно кодовой таблице имеет значения от 0 до 255, из которых первые 32 – управляющие. Для кодирования используется кодовая таблица Win1251.

Управляющие символы (ESC-последовательности) в коде программы требуют двух символов для представления, первый символ – слэш `'\': '\\', '\n', '\t', '\''.`

Операции над символами выполняются над значениями их кодов.

2. Объектный тип `std::string`

Тип `string` в пространстве имен `std` – объектный тип для представления строк, основанный на шаблонах. Входит в стандартную библиотеку C++. Чтобы использовать в коде, нужно включить директиву <`string`>.

`string` как объект имеет:
1) данные – набор данных моделирует символьный массив;
2) методы доступа к данным – функции и операции.

Строка типа `std::string` хранит последовательность символов в однобайтовой кодировке. Признаком окончания строки является управляющий ноль: `'\0'.`

Распределение памяти при инициализации строк и при выполнении методов строк выполняется автоматически. Методы не изменяют существующую строку, а порождают новые строки, поэтому длина строки может изменяться.

Доступ к данным и управление данными осуществляется через методы строк.

Обращение к методам строки выполняется операцией разыменования «.»:

Имя_объекта.Имя_метода(параметры)

Методы строк

- **Объявление и инициализация строк**

При объявлениии строки ее можно инициализировать массивом символов или другим объектом типа `string`. Конструктор строки перегружен, вызывается при объявлении и инициализирует строку.

```
string My_Text;                    // конструктор копии:
string My_Text("Это мой текст");   // получает *char
string New_Text(My_Text);          // получает string
```

- **Определение длины строки.**

```
int Len = str.size();
Len = str.length();
```

- **Присваивание и копирование одной строки в другую.**

Для `string` перегружена операция присваивания = .

```
str2 = str1;     // Копирует  str1 в str2.
```

- **Конкатенация (сцепление) строк.**

Для конкатенации строк перегружены операции сложения + и сложения с присваиванием +=:

```
string str1 = "Hello";
string str2 = "World";
string str3 = str1 + str2; // В строке str3 : "HelloWorld".
```

- **Сравнение строк.**

Для сравнения строк перегружены операции: ==, !=, <, >, <=, >=.

Сравнение выполняется в лексикографическом порядке. Возвращает `bool`.

Метод `compare()` для строк лежит в основе операций, и возвращает `int` (`0`, если строки равны, `-1` или `+1`).

Обращение: `if(str1.compare(str2) == 0))`

.

- **Доступ к отдельным символам строки для чтения и записи.**

Для доступа к отдельным символам строки используются либо операция разыменования [], либо метод `at()`.

1. Операция разыменования [] перегружена:

```
string str = "Hello, World";
cout << str[7] << str[0] << endl;     // получает "oH"
```

2. Метод `at()` требует указать индекс как аргумент функции:

```
string str = "Hello World";
cout << str.at(7) << str.at(0) << endl; // получает "oH".
```

В отличие от [], метод `at()` обеспечивает проверку границ и генерирует исключение `out_of_range` .

- **Преобразование строк: удаление, вставка, замена части строки.**

Группа методов позволяет редактировать строку:

- `append` – добавление строки к строке;
- `assign`– изменение содержания строки;
- `insert`– вставка в строку;
- `erase` – удаление части строки;
- `replace` – замена части строки.

В любом случае параметрами являются точка редактирования и образец.

- **Поиск для строк**

Методы поиска позволяют задать образец поиска и локализовать место вхождения подстроки в строку:

- find – первое вхождение строки в строку;
- rfind – последнее вхождение строки в строку;
- find_first_of – первое вхождение символа в строку;
- find_last_of – последнее вхождение символа в строку.

Возвращают точку: положение образца поиска в строке.

- **Извлечение подстроки**

Метод str.substr(pos, n) – возвращает подстроку исходной строки, начиная с позиции pos и включая n символов, или до конца строки.

- **Преобразование числа к строке и обратно**

Методы string позволяют конвертировать строку в числовое значение, и обратно:

- to_string – преобразует число в строку;
- stoi – преобразует строку в целое число ;
- stof - преобразует строку в вещественное число.

- **Ввод и вывод данных типа string**

Для ввода строк используется объект cin, для вывода – cout. cin вводит значение до пробельного символа. Для ввода всей строки исползуется функция getline().

3. Алгоритмы работы с текстом

В основе редактирования текста лежат алгоритмы работы с данными типов char и string. Практически все алгоритмы, на основе которых можно решить любую задачу обработки текста, реализованы в методах строк.

Как пример рассмотрим алгоритм удаления из строки первого и последнего слов. Примитивный вариант решения, это поиск первого пробела по символам строки, и сдвиг массива влево, затем поиск последнего пробела.

Однако в методах string есть методы поиска и удаления, применение которых приводит к следующему коду:

```
int k = T.find_first_of(' ');
T.erase(0, k);
k = T.rfind(' ');
T.erase(k);
```

Смысл метода понятен из названия, смысл параметров очевиден из логики метода.

4. Функции работы с текстом

В соответствии с принципом модульного стиля программирования, прикладные задачи редактирования текста решаются с использованием функций. Функции работы со строками получают не массив символов, а объект типа string. Следовательно, механизм передачи строки как параметра такой же, как и для переменных: по значению в функцию передается копия строки, по адресу – адрес объекта. Чтобы вернуть результат, функция должна вернуть объект string.

5. Работа с текстовыми файлами

Для работы с файлами необходимо подключить заголовочный файл <fstream>, в котором определены классы ввода и вывода, и подключены заголовочные файлы <ifstream> − файловый ввод и <ofstream> − файловый вывод.

Для записи в файл используется объект ofstream, для чтения – объект ifstream.

На консоль ввод и вывод выполняются с помощью объектов cin и cout. Файловый ввод/вывод полностью аналогичен вводу/выводу на консоль, если создать объекты для переназначения ввода/вывода в файл:

- создать объект класса ofstream/ifstream;

- связать объект класса с файлом, в который будет производиться запись/чтение;
- записать/прочитать данные в файл;
- закрыть файл.

Запись в файл

Для записи в файл к объекту `fstream` применяется оператор `<<` , как и при выводе на консоль:

```
#include <fstream>
…
std::ofstream out;              // out - поток для записи.
out.open("D:\\hello.txt");      // Открыть файл для записи.
if (out.is_open())
   out << "Hello World!" << std::endl;
out.close();                    // Чтобы сохранить.
```

Если физически файл с именем `hello.txt` не существует, он будет создан. Если такой файл есть, данные будут записаны заново.

Чтение из файла

Функция `getline()` принимает поток для чтения и переменную, в которую надо считать текст. Читает одну строку как текст.

```
#include <fstream>
 …
std::string line;
std::ifstream in("D:\\hello.txt");  // Открыть файл для чтения
if (in.is_open()) {
   while (getline(in, line))        // Прочитать строку.
      std::cout << line << std::endl;
   }
in.close();                         // Закрыть файл.
```

Для чтения данных из файла для объектов `ifstream` может применяться оператор `>>` , как и при чтении с консоли:

```
std::ifstream in("D:\\operations.txt");
 …
if (in.is_open()) {
   for (int i = 0; i <Count ;i++)
      in >> Arr[i];
    }
in.close();
```

Задание

Заголовочный файл `My_string.h` скопируйте в папку проекта и присоедините к проекту, откройте и ознакомьтесь с содержимым.

Упражнение 1. Строка как параметр функции

В примере функции F1 и F2 должны удалить из строки первое и последнее слова. Обе функции получают строку F1 по значению. Строка передается в функцию как объект, значит, в теле функции создается копия строки.

Опишите и инициализируйте строку, передайте в функцию F1, затем в функцию F2. Изучите механизм получения новой строки. Измените F1 так, чтобы она получала и изменяла строку-параметр.

Упражнение 2. Доступ к символам строки

Опишите и инициализируйте строку, передайте в функцию `remove_E`. Для изменения строки функция напрямую обращается к методу `at()`.

Перепишите функцию так, чтобы она не заменяла символы, а вставляла разделитель `" ! "` после каждого пятого символа.

Упражнение 3. Извлечение фрагментов строк, массивы строк

Обратитесь к функции `Dictionary`, получающей словарь текста – массив слов, содержащихся в строке.

Измените функцию так, чтобы в словарь не включались бы повторяющиеся слова.

Выводы

1. `String`, это объект. Строку как объект можно передавать либо по значению, либо по адресу. Когда в списке формальных параметров функции строка, то функция получает копию.

2. Если функция изменяет строку или формирует новую, она должна вернуть в `main` объект типа `string`.

3. При вызове функции, получающей новые данные, на стороне `main` должны быть объявлены объекты, принимающие эти значения.