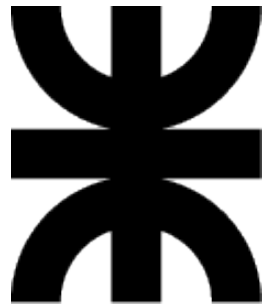
Informática II
Cadenas en C++, `std::string`

Gonzalo F. Perez Paina



Universidad Tecnológica Nacional
Facultad Regional Córdoba
UTN-FRC

– 2024 –

Definición e inicialización

- Una variable tipo cadena se almacena como una secuencia de letras u otros caracteres, tal como `"Hola"` o `"GNU's Not Unix!"`.

Definición e inicialización

- Una variable tipo cadena se almacena como una secuencia de letras u otros caracteres, tal como `"Hola"` o `"GNU's Not Unix!"`.
- Al igual que otros tipos de datos, para crear una cadena `std::string` se puede definir y luego asignarle un valor.

Definición e inicialización

- Una variable tipo cadena se almacena como una secuencia de letras u otros caracteres, tal como `"Hola"` o `"GNU's Not Unix!"`.
- Al igual que otros tipos de datos, para crear una cadena `std::string` se puede definir y luego asignarle un valor.

Definición y asignación:

```
std::string cadena;
cadena = "Esto es una cadena.";
```

Definición e inicialización

- Una variable tipo cadena se almacena como una secuencia de letras u otros caracteres, tal como `"Hola"` o `"GNU's Not Unix!"`.
- Al igual que otros tipos de datos, para crear una cadena `std::string` se puede definir y luego asignarle un valor.

Definición y asignación:

```
std::string cadena;
cadena = "Esto es una cadena.";
```

o bien, combinando ambas sentencias, se define e inicializa la variable

```
std::string cadena = "Esto es una cadena.";
```

Definición e inicialización

- Una variable tipo cadena se almacena como una secuencia de letras u otros caracteres, tal como `"Hola"` o `"GNU's Not Unix!"`.
- Al igual que otros tipos de datos, para crear una cadena `std::string` se puede definir y luego asignarle un valor.

Definición y asignación:

```
std::string cadena;
cadena = "Esto es una cadena.";
```

o bien, combinando ambas sentencias, se define e inicializa la variable

```
std::string cadena = "Esto es una cadena.";
```

Para mostrar una cadena por la salida estándar:

```
std::cout << cadena << std::endl;
```

Definición e inicialización

Para utilizar el tipo de dato `string` es necesario incluir el archivo de cabecera `string` en un programa de C++

Definición e inicialización

Para utilizar el tipo de dato `string` es necesario incluir el archivo de cabecera `string` en un programa de C++

```
1  #include <iostream>
2  #include <string>
3
4  using namespace std;
5
6  int main()
7  {
8    string cadena;
9    cadena = "Esto es una cadena.";
10
11   cout << cadena << endl;
12   return 0;
13 }
```

Longitud de la cadena

- La función miembro `length()` devuelve la longitud de la cadena (incluyendo espacios y signos de puntuación).
- Para llamar a la función miembro se utiliza el *operador punto*.

Longitud de la cadena

- La función miembro `length()` devuelve la longitud de la cadena (incluyendo espacios y signos de puntuación).
- Para llamar a la función miembro se utiliza el *operador punto*.

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main()
6  {
7    string corta = "Cadena corta.";
8    string larga = "Esta, como puede observarse, \
9                    es una cadena más elaborada.";
10
11   cout << "La cadena corta tiene: " << corta.length()
12     << " caracteres." << endl;
13   cout << "La cadena larga tiene: " << larga.length()
14     << " caracteres." << endl;
15
16   return 0;
17 }
```

Acceso a caracteres

- Se puede acceder a los caracteres individuales de una cadena con el operador `[]`.
- La posición dentro dentro de la cadena `str` se numera desde 0 hasta `str.length() - 1`.

Acceso a caracteres

- Se puede acceder a los caracteres individuales de una cadena con el operador `[]`.
- La posición dentro dentro de la cadena `str` se numera desde 0 hasta `str.length() - 1`.

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main()
6  {
7    string test;
8    test = "El caracter Y pertendce a esta cadena.";
9
10   char ch = test[12]; // ch es 'Y'
11   test[20] = 'e'; // Se corrige el error
12
13   cout << test << endl;
14   cout << "ch = " << ch << endl;
15   return 0;
16 }
```

Acceso a caracteres

- Hay que tener cuidado de no acceder a elementos fuera de los límites de la cadena.
- El operador `[]` no verifica el rango del índice.
- La función miembro `at(int index)` sí verifica el rango.

Acceso a caracteres

- Hay que tener cuidado de no acceder a elementos fuera de los límites de la cadena.
- El operador `[]` no verifica el rango del índice.
- La función miembro `at(int index)` sí verifica el rango.

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main()
6  {
7    string str = "Cadena de 24 caracteres.";
8
9    cout << "Longitud: " << str.length() << endl;
10   cout << "Accediendo al elemento 30 con []" << endl;
11   cout << " el elemento 30 es: " << str[30] << endl;
12
13   cout << "Accediendo al elemento 30 con at()" << endl;
14   cout << " el elemento 30 es: " << str.at(30) << endl;
15   return 0;
16 }
```

Asignación y uso en funciones

- Las cadenas de C++ están diseñadas para tener un comportamiento igual a un tipo de dato primitivo en relación a la asignación.
- La asignación de una cadena a otra realiza una copia de la secuencia de caracteres.

Asignación y uso en funciones

- Las cadenas de C++ están diseñadas para tener un comportamiento igual a un tipo de dato primitivo en relación a la asignación.
- La asignación de una cadena a otra realiza una copia de la secuencia de caracteres.

```
string str1 = "hola";
string str2 = str1; // Realiza una copia
str1[0] = 'H'; // Modifica str1 pero no str2
```

Asignación y uso en funciones

- Las cadenas de C++ están diseñadas para tener un comportamiento igual a un tipo de dato primitivo en relación a la asignación.
- La asignación de una cadena a otra realiza una copia de la secuencia de caracteres.

```
string str1 = "hola";
string str2 = str1; // Realiza una copia
str1[0] = 'H'; // Modifica str1 pero no str2
```

- El pasaje y retorno de cadenas en funciones se realiza por copia.
- Si se modifica un parámetro de cadena dentro de una función, los cambios no se ven en la función llamadora a menos que se haya pasado específicamente la cadena por referencia.

Comparación entre cadenas

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

int main() {
  string nombreUsuario, miNombre = "Juan";

  while(true) {
    cout << "Ingrese su nombre (o 'salir' para finalizar): ";
    getline(cin, nombreUsuario);

    if(nombreUsuario == "Juan") { // Operador '=='
      cout << "Hola Juan! Bienvenido!" << endl;
    } else if(nombreUsuario == "salir") {
      cout << endl;
      break;
    } else if(nombreUsuario != miNombre) { // Operador '!='
      cout << "Hola, " << nombreUsuario << endl;
    } else
      cout << "Ah, sos vos, " << miNombre << endl;
  }
  return 0;
}
```

Concatenación de cadenas

- Si se tiene una cadena `s1` y otra `s2` se puede crear otra a partir de ellas (`s1 + s2`).
- También se puede añadir otra cadena al final con el operador `+=` (o un caracter).

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main()
6  {
7    string nombre = "Gonzalo";
8    string apellido = " Perez";
9
10   string nombreCompleto = nombre + apellido;
11   nombreCompleto += " Paina"; // Añade otra cadena
12   nombreCompleto += "."; // Añade un caracter
13
14   cout << nombre << apellido << endl;
15   cout << nombreCompleto << endl;
16   return 0;
17 }
```

Búsqueda en cadenas

- Para buscar una cadena o caracter se utiliza la función miembro `find()`.
- Por ejemplo `str.find(key)` busca `key` en `str`.
- El parámetro `key` puede ser una cadena o un caracter. (por qué?).
- El valor de retorno puede ser la posición de inicio donde se encuentra `key` o bien la constante `std::npos` que indica que no se encontró `key`.
- En ocaciones es necesario controlar en qué parte de la cadena se desea buscar.
- El segundo argumento que es opcional de `find()` indica la posición de inicio, si no se utiliza se asume como 0.

Búsqueda en cadenas

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

int main() {
  string sentence = "All the world's a stage, "\
                    "And all the men and women merely players";

  int firstMe = sentence.find("me");
  int secondMe = sentence.find("me", firstMe + 1);
  int thirdMe = sentence.find("me", secondMe + 1);
  int gPos = sentence.find('g');
  int zPos = sentence.find('z'); // Returns string::npos

  cout << "Primer 'me': " << firstMe << endl;
  cout << "Segundo 'me': " << secondMe << endl;
  cout << "Tercer 'me: " << thirdMe << endl;

  cout << "Está 'g' en la cadena? ";
  cout << (gPos != string::npos ? "Si!" : "No!") << endl;
  cout << "Está 'z' en la cadena? ";
  cout << (zPos != string::npos ? "Si!" : "No!") << endl;
  return 0;
}
```

Búsqueda en cadenas

```
$> /string_find
Primer 'me': 37
Segundo 'me': 47
Tercer 'me: 51
Está 'g' en la cadena? Si!
Está 'z' en la cadena? No!
```

Cadena al estilo C

```
1  #include <iostream>
2  #include <string>
3  #include <fstream>
4
5  using namespace std;
6
7  int main()
8  {
9    ifstream fs;
10   string filename = "info2.txt";
11   string s;
12
13   // La función para abrir el archivo necesita una cadena
14   // al estilo-C
15   fs.open(filename.c_str());
16
17   if(fs.fail()) return -1; // No se pudo abrir el archivo
18
19   // Procesar el archivo;
20
21   fs.close();
22   return 0;
23 }
```