



Tema 2

Cadenas en C

La clase `string` en C++

Definición
Declaración
Entrada/salida
Métodos
Operaciones

Conversiones

Comparativa

Ejercicios

Tema 2: La clase `string` Programación 2

Curso 2013-2014

Notas



Tema 2

Cadenas en C

La clase `string` en C++

Definición
Declaración
Entrada/salida
Métodos
Operaciones

Conversiones

Comparativa

Ejercicios

Índice

- 1 Cadenas de caracteres en C
- 2 La clase `string` en C++
 - Definición
 - Declaración
 - Entrada / salida
 - Métodos
 - Operaciones
- 3 Conversiones entre vectores de caracteres y `string`
- 4 Comparativa
- 5 Ejercicios

Notas

Cadenas de caracteres en C

- En C, las cadenas de caracteres tienen tamaño fijo

```
char cad[10];
```

- También es posible declararlas con tamaño dinámico de diferentes formas:

```
int tamCad;
cin >> tamCad;
char cad[tamCad]; // C++, no recomendado por estandar
```

```
char *cad = (char *)malloc(tamCad*sizeof(char)); // C
```

```
char *cad = new char[tamCad]; // C++
```

- Sin embargo, una vez declaradas no pueden cambiar de tamaño.

Notas

Entrada/salida con cadenas de C desde C++

- cout : como el resto de variables simples (int, float,...) **OJO:** la cadena debe terminar en \0
- cin : **casi** como el resto de variables
 - cin ignora los blancos antes de la cadena ...
 - Termina de leer en cuanto encuentra el primer blanco (espacio, tabulador o \n) después de la cadena.
Problema: ¿y si la cadena tiene blancos?
 - No limita el número de caracteres que se leen.
PROBLEMA GORDO: ¿y si lo que se lee no cabe en el vector?

Notas

getline

- Con `getline` se pueden leer cadenas con blancos y controlar su tamaño

```
cin.getline(cadena, TAM)
```

- Lee como mucho TAM-1 caracteres o hasta que llegue al final de la línea
- El '\n' del final de la línea se lee pero no se mete en la cadena. La función añade '\0' al final de lo que ha leído (por eso sólo lee TAM-1 caracteres)
- ... pero si el usuario introduce más caracteres de los que caben, estos se quedan en el buffer y la siguiente lectura falla.

Notas

Problemas mezclando » con getline

```
int num;  cout << "Num= ";  
cin >> num;  char cadena[1000];  
cout << "Escribe una cadena: " ;  
cin.getline(cadena, 1000);  
cout << "Lo que he leído es: " << cadena << endl;
```

```
Num= 10  
Escribe una cadena: Lo que he leído es:
```

¿Por qué?

- Con '>>' se lee el 10, pero se deja de leer cuando se encuentra el primer carácter no numérico: '\n'
- `getline` encuentra en el buffer un '\n', con lo que lee una cadena vacía.

Solución:

```
cin >> num;  
cin.get(); // Para leer el \n
```

Notas

Funciones de `string.h` (1/2)

- `strlen` devuelve la longitud de una cadena.

```
char cad[] = "adios";  
cout << strlen(cad) << endl; // Imprime 5
```

- `strcmp` compara dos cadenas en orden lexicográfico, devolviendo un valor negativo si $a < b$, cero si $a == b$, o positivo si $a > b$.

```
char cad[] = "adios"; char otra[] = "adeu";  
cout << strcmp(cad, otra) << strcmp(otra, cad)  
    << strcmp(cad, cad) << endl; // Imprime -1 -1 0
```

- `strcpy` copia una cadena en otra (ojo, violación de segmento si no cabe)

```
char cad[10];  
strcpy(cad, "hola"); // cabe, son 4 + \0 = 5 caracteres
```

Notas

Funciones de `string.h` (2/2)

- Las funciones `strncpy`, `strncpy` comparan o copian sólo los n primeros caracteres. Ejemplo: Ejemplo:

```
char cad[tCAD];  
strncpy(cad, "hola, mundo", 4); // solo copia "hola"  
cad[4] = '\0'; // Al contrario que strcpy, no copia el \0
```

- Para pasar una cadena de caracteres a entero o float, se puede usar `atoi` o `atof`.

```
#include <cstdlib> // Importante!  
  
char cad[]="100";  
int n=atoi(cad); // n vale 100  
  
char cad2[]="10.5";  
float f=atof(cad2); // f vale 10.5
```

Notas



Tema 2

Cadenas en C

La clase `string` en C++

Definición

- Declaración
- Entrada/salida
- Métodos
- Operaciones

Conversiones

Comparativa

Ejercicios

Strings en C++: Definición

- En C++ se puede usar la clase `string` (aunque también se pueden usar los vectores de caracteres)
- En el paso de parámetros (valor / referencia), se parece a cualquier tipo simple (`int`, `float`, ...),
- Es de tamaño variable (no hay máximo), y puede crecer
- No termina en `\0`

Notas



Tema 2

Cadenas en C

La clase `string` en C++

Declaración

- Entrada/salida
- Métodos
- Operaciones

Conversiones

Comparativa

Ejercicios

Declaración

- Constante:

```
const string cadena="hola";
```
- Variable:

```
string cadena;
```
- Declaración con inicialización:

```
string cadena = "hola";
```

Notas

Entrada/salida con `strings`

- `cout` : como los tipos simples (`int`, `float`, ...)
- `cin` : casi como con vectores de caracteres
 - Ignora los blancos antes de la cadena y termina de leer en cuanto encuentra el primer blanco después de la cadena.
 - Si la cadena tiene blancos :

```
getline(cin, cadena)
```

(ojo, cambia la sintaxis con respecto a los vectores de caracteres)

- No limita el número de caracteres que se leen. Con `strings` no hay problema!!
- OJO: si se hace `'>>'` y después `'getline'` ocurre lo mismo que con vectores de caracteres

Notas

Métodos de `string` (1/2)

Al ser una clase, los métodos se invocan con un punto tras el nombre de la variable. Ejemplo:

```
unsigned int tam=s.length(); // s es un string
```

- Longitud del string:

```
unsigned int length();
```

- Búsqueda de subcadena:

```
unsigned int find (const string str, unsigned int pos=0);  
// Si no se encuentra devuelve la constante string::npos
```

- Reemplazo de subcadena:

```
string& replace (unsigned int pos, unsigned int tam,  
               const string str);
```

- Quitar una subcadena

```
string& erase (unsigned int pos=0, unsigned int tam=npos);
```

Notas

Métodos de string (2/2)

```
string a="Hay una taza en esta cocina con tazas";
string b="taza";

// Longitud de a
unsigned int tam = a.length();

// Buscamos la primera palabra 'taza'
unsigned int encontrado=a.find(b);
if (encontrado!=string::npos)
    cout << "primera 'taza' en: " << encontrado << endl;
else cout << "palabra 'taza' no encontrada";

// Buscamos la segunda palabra 'taza'
encontrado=a.find("taza",encontrado+b.length());
if (encontrado!=string::npos)
    cout << "segunda 'taza' en: " << encontrado << endl;
else cout << "palabra 'taza' no encontrada";

// Sustituimos la primera 'taza' por 'botella'
a.replace(a.find(b),b.length(),"botella");
cout << a << endl;
```

Notas

Operaciones de string

- Comparaciones: ==, !=, >, <, >= y <=
- Asignación de una cadena a otra: con el operador =, como con cualquier tipo simple.
- Concatenación de cadenas: con el operador +

```
s1 = "hola" ; s2 = "mundo";
s = s1 + ", " + s2;
cout << s << endl ; // sale 'hola, mundo'
```

- Acceso a componentes como si fuera un vector de caracteres: **solamente si el string tiene algo.**

```
s = "hola" ; car = s[3]; s[0] = 'H';
cout << s << ":" << car << endl ; // sale 'Hola:a'
```

```
// Ejemplo de recorrido
for (unsigned int i=0; i<s.length(); i++)
    s[i]='f';
```

Notas

Conversión entre vectores de caracteres y string

- vector de caracteres a string : con la asignación (=)

```
char cad[] = "hola";
string s ;
s = cad ; s = s + ", mundo";
```

- string a vector de caracteres : con c_str

```
char cad[tCAD];
string s = "mundo";

// OJO: debe haber sitio suficiente en 'cad'
strcpy(cad, s.c_str());
```

Notas

Conversión entre string y enteros

- Entero a string

```
#include <sstream>

int n=100;
stringstream ss;

ss << n;
// Tambien se pueden concatenar mas cosas, por ejemplo:
// ss << "El numero es: " << n << endl;

string numero=ss.str();
```

- string a entero

```
string numero="100";
int n=atoi(numero.c_str());
```

Notas



Tema 2

Cadenas en C

La clase
string en
C++

Definición
Declaración
Entrada/salida
Métodos
Operaciones

Conversiones

Comparativa

Ejercicios

Comparativa entre vectores de caracteres y string

vectores de caracteres	string
<pre>char cad[TAM]; char cad[]="hola";</pre>	<pre>string s; string s="hola";</pre>
<pre>strlen(cad) cin.getline(cad,TAM); if (!strcmp(c1,c2)){.. strcpy(c1,c2); strcat(c1,c2);</pre>	<pre>s.length() getline(cin,s); if (s1 == s2){.. s1 = s2; s1 = s1 + s2;</pre>
<pre>strcpy(cad,s.c_str());</pre>	<pre>s = cad;</pre>
Terminan con '\0' Tamaño reservado fijo Tamaño ocupado variable	NO terminan con '\0' El tamaño reservado puede crecer Tamaño ocupado = tamaño reservado
Se usan en ficheros binarios	NO se pueden usar en ficheros binarios

Notas



Tema 2

Cadenas en C

La clase
string en
C++

Definición
Declaración
Entrada/salida
Métodos
Operaciones

Conversiones

Comparativa

Ejercicios

Ejercicios (1/4)

Ejercicio 1 Diseña una función “SubCadena” que extraiga la subcadena de longitud n que empieza en la posición p de otra cadena. Tanto el argumento como el valor de retorno deben ser `string`.

`SubCadena("hooooola", 2, 5) ⇒ "la"`

Ejercicio 2

Diseña una función llamada “BorraCaracterDeCadena” que dados un `string` y un carácter borre todas las apariciones del carácter en el `string`.

`"hola, mundo" 'o' ⇒ "hla, mund"`

Notas



Ejercicios (2/4)

Tema 2

Cadenas en C

La clase
string en
C++

Definición
Declaración
Entrada/salida
Métodos
Operaciones

Conversiones

Comparativa

Ejercicios

Ejercicio 3

Diseña una función “`BuscarSubCadena`” que busque la primera aparición de una subcadena `a` dentro de una cadena `b`, y devuelva su posición o `-1` si no está. Tanto `a` como `b` deben ser `string`.

`BuscarSubCadena("ool", "hoolola") ⇒ 2`

Ampliaciones:

- 1 Ampliar la función para que admita otro parámetro que sea el número de aparición (si vale 1 sería como la función original)
- 2 Hacer otra función similar que devuelva el número de apariciones de la subcadena en la cadena.

Notas



Ejercicios (3/4)

Tema 2

Cadenas en C

La clase
string en
C++

Definición
Declaración
Entrada/salida
Métodos
Operaciones

Conversiones

Comparativa

Ejercicios

Ejercicio 4

Diseña una función “`Codifica`” que codifique una cadena sumando una cantidad `c` al código ASCII de cada carácter, pero teniendo en cuenta que el resultado debe ser una letra.

Por ejemplo, si $n = 3$, la ‘a’ se codifica como ‘d’, la ‘b’ como ‘e’, ..., la ‘x’ como ‘a’, la ‘y’ como ‘b’ y la ‘z’ como ‘c’.

La función debe admitir letras mayúsculas o minúsculas, los caracteres que no sean letras no se deben codificar y el argumento debe ser `string`.

`Codifica("hola, mundo", 3) ⇒ "krod, pxqgr"`

Notas



Tema 2

Cadenas en C

La clase
string en
C++

Definición
Declaración
Entrada/salida
Métodos
Operaciones

Conversiones

Comparativa

Ejercicios

Ejercicios (4/4)

Ejercicio 5

Diseña una función “EsPalindromo” que devuelva `true` si el `string` que se le pasa como parámetro es palíndromo.

`EsPalindromo("hola,aloh") ⇒ true`

`EsPalindromo("hola, aloh") ⇒ false`

Ejercicio 6

Diseña una función llamada “CreaPalindromo” que añada a un `string` el mismo `string` invertido, de forma que el resultado sea un palíndromo.

`"hola, mundo" ⇒ "hola, mundoodnum ,aloh"`

Existen dos formas de hacerlo:

- `string CreaPalindromo(string);`
- `void CreaPalindromo(string &);`

Notas

Notas
