Fundamentos de Programación
Sesión 8

Actividades a realizar en casa

Actividad: Resolución de problemas.

Resolved los siguientes problemas de la relación 3. Recordad que, **ANTES** del inicio de esta sesión en el aula de ordenadores, hay que subir las soluciones a PRADO.

5 (Intercambiar componentes)
6 (Subvector)
7 (Eliminar mayúsculas)
8 (Eliminar mayúsculas 2)
9 (Criba de Eratóstenes)
12 (Subsecuencias)

Actividades a realizar en las aulas de ordenadores

Cadenas de caracteres tipo C (cstring)

Para el tratamiento de cadenas de caracteres (por ejemplo, para trabajar con datos como nombres, apellidos, direcciones, etc.), en C++ contamos con dos tipos de datos:

- cstring: cadena de caracteres heredada de C.
- string: cadena de caracteres propia de C++ (que es la que estudiamos en FP).

En este guion trataremos las cadenas del primer tipo.

Definición. Una cadena de caracteres tipo C es simplemente un vector de caracteres (un array, de los que hemos visto en el Tema 3 de teoría) que tiene la particularidad de que termina con un carácter especial, que indica el final de la cadena, llamado carácter fin de cadena o carácter nulo. El carácter fin de cadena tiene el código ASCII 0 (es el primero de la tabla ASCII) y se representa por '\0'. Este símbolo sirve como valor *centinela* para marcar el final de las componentes útiles del vector (tal y como hemos visto en el Tema 3 de teoría, en la subsección 1.3.2 "Gestión de componentes utilizadas", página 305).

```
char apellido[20] ={'N','a','v','a','r','r','o','\0'};
```

Literales. Recordemos que un *literal de cadena de caracteres* es una secuencia de cero o más caracteres encerrados entre comillas dobles. Por ejemplo,

```
"Hola Mundo"
```

El compilador considera que esto es un array de char (constante) con un tamaño igual a su longitud más uno (para el carácter nulo). Por ejemplo,

- "Hola" de tipo const char[5]
- "Hola mundo" de tipo const char[11]
- "" de tipo const char[1]

Declaración e inicialización. Para declarar e inicializar una cadena de caracteres tipo C, se utiliza la misma sintaxis que para los vectores.

```
char nombre[10] ={'J','a','v','i','e','r','\0'};
```

'J'	'a'	'v'	'i'	'e'	'r'	'\0'	?	?	?

```
char nombre[] ={'J','a','v','i','e','r','\0'}; // Asume char[7]
```

Equivalente a las anteriores son:

```
char nombre[10]="Javier";
char nombre[]="Javier";
```

NOTA: Observad que si inicializamos letra a letra, necesitamos insertar el símbolo de final de cadena nosotros mismo. Si inicializamos utilizando un literal de cadena de caracteres, no es necesario. Es decir,

```
char cadena[]="Hola";    // es un char[5], es {'H', 'o', 'l', 'a', '\0'}
char cadena[]={'H','o','l','a'};   // es un char[4]
```

Escritura y lectura. Para leer y escribir cadenas se pueden usar las operaciones de lectura y escritura ya conocidas, como cin o cout

```
1 #include<iostream>
2 using namespace std;
3
4 int main(){
5    char nombre[80];
6    cout << "Introduce tu nombre: ";
7    cin >> nombre;
8    cout << "El nombre introducido es: " << nombre;
9 }
```

CUIDADO: cin salta separadores antes del dato y se detiene cuando encuentra un separador (saltos de línea, espacios en blanco y tabuladores). Es decir, no debe usarse para leer cadenas de caracteres que contengan espacios en blanco. Además, no consume el separador, que quedará pendiente para próximas operaciones de lectura. Para insertar espacios en blanco, se puede utilizar getline

cin.getline(cadena, tamaño);

Lee hasta que se encuentra un salto de línea o se alcanza el límite de lectura (tamaño caracteres).

Conversión. Podemos hacer fácilmente la conversión entre cadenas cstring y string:

- Para pasar de un cstring a un string, el propio compilador realiza una conversión (casting)
- Para pasar de string a cstring, debemos utilizar la función c_str() de la biblioteca string.

```
 6 int main(){
 7    char cadena1[]="Hola";
 8    string cadena2;
 9    char cadena3[10];
10
11    cadena2=cadena1; // cstring-->string
12    strcpy (cadena3, cadena2.c_str()); // string-->cstring
13    cout<<"cadena2="<<cadena2<<endl;
14    cout<<"cadena3="<<cadena3<<endl;
15 }
```

Manejo. La biblioteca cstring proporciona funciones de manejo de cadenas de caracteres de C. Algunos ejemplos de funciones son:

- strcpy(cadena1, cadena2). Copia cadena2 en cadena1. Es el operador de asignación de cadenas (recordad que no es posible asignar vectores).
- strlen(cadena). Devuelve la longitud de cadena.
- strcat(cadena1, cadena2). Concatena cadena1 al final de cadena2 y el resultado se almacena en cadena1.
- strcmp(cadena1, cadena2). Compara las cadenas cadena1 y cadena2. Si la cadena cadena1 es menor (lexicográficamente) que cadena2 devuelve un valor menor que cero, si son iguales devuelve 0 y en otro caso devuelve un valor mayor que cero.

Copia, compila y ejecuta el siguiente código:

```
#include<iostream>
#include<cstring>
using namespace std;
int main(){
    const int DIM=100;
    char c1[DIM]="Hola", c2[DIM];
    strcpy(c2, "mundo");
    strcat(c1, " ");
    strcat(c1, c2);
    cout <<"Longitudes:"<<strlen(c1)<<" "<<strlen(c2);
    cout << "\nc1: " << c1 << " c2: " << c2;
    if (strcmp(c1,"adiós mundo cruel") < 0)
        cout << "\nCuidado con las mayúsculas\n";
    if (strcmp(c2, "mucho") > 0)
        cout << "\n\"mundo\" es mayor que \"mucho\"\n";
}
```

Actividad: Resolución de problemas.

Resolved los siguientes problemas de la relación 3.

19 (Subcadena)
20 (Insercción de cadenas)
11 (TipoConjunto)