CLASE STRING

CLASE STRING (Cadenas de caracteres)

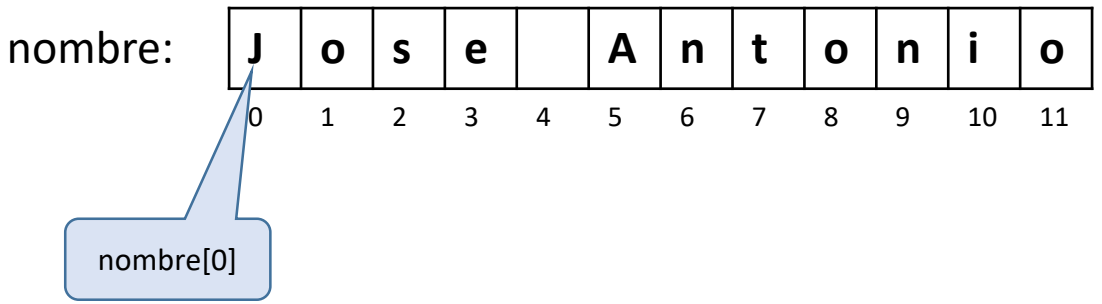
Representan cadenas de caracteres de longitud finita. Para utilizar el tipo string es necesario incluir la biblioteca estándar <string>. Valor por defecto la cadena vacía ("")

```
//string01.cpp

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

int main()
{
    string nombre = "Jose Antonio" ;
    string apellidos("Ruiz"); //Otra forma de inicializar
    cout << nombre << endl;
    cout << apellidos << endl;
    cout << nombre[0]<<endl;

}
```



El operador de entrada (>>) se comporta de la siguiente forma: elimina los espacios en blanco que hubiera al principio de la entrada de datos, y lee dicha entrada hasta que encuentre algún carácter de espacio en blanco, que no será leído y permanecerá en el buffer de entrada, hasta la próxima operación de entrada.

```
//string01b.cpp

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

int main()
{
        string nombre = "Jose Antonio" ;
        cout << nombre << endl;
        cout << nombre[0]<<endl;
        cout << "Introduzca un nombre:";
        cin >> nombre;
        /* Si introducimos “Jose Antonio” nombre toma el valor Jose y queda “ Antonio<enter>” en buffer de entrada.
        por tanto el siguiente cin tomará “Antonio” y aceptará sin dar enter */
        cout<<endl<<nombre;
        cin >> nombre;
        cout<<endl<<nombre;
        return 0;

}
```

Si se desea leer una secuencia de caracteres que incluya espacios en blanco, utilizaremos la función **getline** en lugar del operador >>.

La función **getline** lee y almacena en una variable de tipo string todos los caracteres del buffer de entrada, hasta leer el carácter de fin de línea (ENTER), sin eliminar los espacios iniciales.

getline(cin, var_string)

```
//string03.cpp
#include <iostream>
#include <string>
using namespace std ;
int main()
{
  string nombre ;
  int edad ;
  for (int i = 0; i < 5; ++i) {
                cout << "Introduzca el nombre: " ;
                getline(cin, nombre) ;
                cout << "Introduzca edad: " ;
                cin >> edad ;
                cout << "Edad: " << edad << " Nombre: [" << nombre << "]" << endl ;
                }
  return 0;
}
```

¡OJO!

Funcionamiento inadecuado. Después de introducir edad se **queda <enter> en buffer** y saltará la ejecución de cada getline().

Para vaciar el buffer de entrada se usará la instrucción **cin >> ws**

```
//string04.cpp
#include <iostream>
#include <string>
using namespace std ;
int main()
{
		string nombre ;
		int edad ;
		for (int i = 0; i < 5; ++i) {
				cout << "Introduzca el nombre: " ;
				cin >> ws;  //elimina caracteres del buffer de entrada
				getline(cin, nombre) ;
				cout << "Introduzca edad: " ;
				cin >> edad ;
				cout << "Edad: " << edad << " Nombre:" << nombre  << endl ;
		}
}
```

Operadores y métodos cadenas

Asignación (=)

Comparaciones lexicográficas (==, !=, >, >=, <, <=)

Concatenación de cadenas y de caracteres (+, +=)

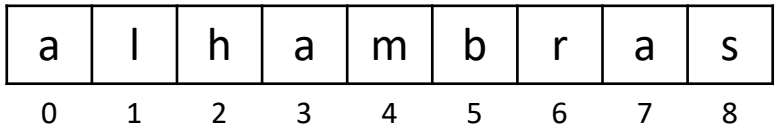
MÉTODOS CLASE STRING

var_string.size()	Longitud de la cadena
var_string.substr(i , n)	Función subcadena
var_string.empty()	True si está vacía.
var_string1.swap(var_string2)	Intercambia valores en cadenas.
var_string.find(string)	Busca una cadena en otra. Si no la encuentra devuelve *string::npos*
var_string.find_first_of(string)	Posición donde primero aparece la subcadena.
var_string.find_last_of(string)	Última posición donde aparece la subcadena.
var_string.find_first_not_of(string)	Posición del primer carácter distinto de subcadena
var_string.find_last_not_of(string)	Posición del último carácter distinto de subcadena

```
//string05.cpp
#include <iostream>
#include <string>
using namespace std ;
int main()
{
        string nombre = "Jose" ;
        string apellido = "Ruiz" ;
        string nombre_completo;

        nombre_completo=nombre+" "+apellido; //Concatenación
        cout<<nombre_completo<<endl;
        cout<<nombre[2]<<endl; //tercer caracter
        cout<<nombre_completo.substr(2,4)<<endl; //subcadena de 4 caracteres
        cout<<nombre_completo.size(); //tamaño de la cadena
        return 0;
}
```

//string05b.cpp




```
#include <iostream>
#include <stdlib.h>
#include <string>
using namespace std ;

int main()
{
        string nombre = "alhambras" ;
        string letras ="a";
        cout<<nombre.find("j")<<endl;   //Devuelve string::npos es igual al mayor valor unsigned int
        cout<<nombre.find_first_of(letras)<<endl;   // devuelve 0
        cout<<nombre.find_last_of(letras)<<endl;   //devuelve 7
        cout<<nombre.find_first_not_of(letras)<<endl;   //devuelve 1
        cout<<nombre.find_last_not_of(letras)<<endl;   //devuelve 8
        system("pause");
        return 0;

}
```

```
//string06.cpp Palindroma
#include <iostream>
#include <string>
using namespace std ;
int main()
{
        string frase_01, frase_02="", frase_03="";
        cout<<"INTRODUZCA FRASE:";
        getline(cin,frase_01);
        for (int i=0 ; i<frase_01.size() ; i++)
                if (frase_01[i]!=32) frase_02=frase_02 + frase_01[i]; //eliminar espacios

        for (int i=0 ; i<frase_02.size() ; i++)
                frase_03=frase_02[i]+ frase_03;

        cout<<endl<<frase_02;
        cout<<endl<<frase_03<<endl;
        if (frase_02 == frase_03)
                cout<<"palindroma";
        else
                cout<<" no es palindroma";
        return 0;
}
```

```
//JUEGO AHORCADO
#include <iostream>
#include <stdlib.h>
#include <string>
using namespace std ;
int main()
{
		string frase = "alhambras" ;
		string ahorcado;
		char letra;
		int i,intentos;

		ahorcado=""; intentos=0;
		for (i=0; i<frase.size(); i++) ahorcado=ahorcado+"-"; //string ahorcado con tantos guiones como frase;
		do
		{
			cout<<ahorcado<<endl;
			do
			{
				cout<<"adivina una letra:";
				cin>>letra;
				intentos++;
				i=frase.find(letra);
			}
			while(i==string::npos);  //mientras que no adivine uma letra
			for (i=0; i<frase.size(); i++)
				if (frase[i]==letra) ahorcado[i]=letra;  //cambiamos guiones por el carácter adivinado
		}
		while(ahorcado != frase);  //mientras que los dos string sean distintos

		cout<<"intentos="<<intentos<<endl;
		cout<<ahorcado<<endl;
		system("pause");
		return 0;

}
```

Registros o Estructuras

Un registro representa un valor compuesto por un número determinado de elementos, que pueden ser de distintos tipos (simples y compuestos).

```
struct <id_estructura>
{
        <tipo_01>  <id_campo1>;
        <tipo_02>  <id_campo2>;
        ……
        <tipo_n>  <id_campon>;
} ;

<id_estructura> <id_var>;
```

```
struct fecha
{
        unsigned dia ;
        unsigned mes ;
        unsigned anio ;
} ;

fecha  fecha_nac; //Una vez definida podemos crear variables de este tipo
```

Una vez declarada una entidad (constante o variable) de tipo registro, por ejemplo la variable fecha_nac, podemos referirnos a ella en su globalidad (realizando asignaciones y pasos de parámetros) o acceder a sus componentes (campos) utilizando el operador punto (.)

fecha_nac.dia

```
//registro01.cpp
#include <iostream>
using namespace std;
int main()
{
        struct fecha
        {
                unsigned dia ;
                unsigned mes ;
                unsigned anio ;
        } ;
        fecha agenda[5];
        for (int i=0;i<5;i++)
        {
                cout<<"Fecha "<<i<<endl;
                cout<<"Dia:";cin >>agenda[i].dia;
                cout<<"Mes:";cin >>agenda[i].mes;
                cout<<"Año:";cin >>agenda[i].anio;
        }
        system("cls");
        for (int i=0;i<5;i++)
                cout<<agenda[i].dia<<"/"<<agenda[i].mes<<"/"<<agenda[i].anio<<endl;
        return 0;
 }
```