



A decorative graphic on the left side of the slide, consisting of a network of white lines and small circles on a dark blue background, resembling a circuit board or a neural network.

MANEJO DE CADENAS

CATEDRÁTICO: ING. DAVID RAJO



OBJETIVOS

- Crear programas que permitan E/S de datos tipo cadena.
 - Manipulación de cadenas.
- 
- 

INTRODUCCIÓN

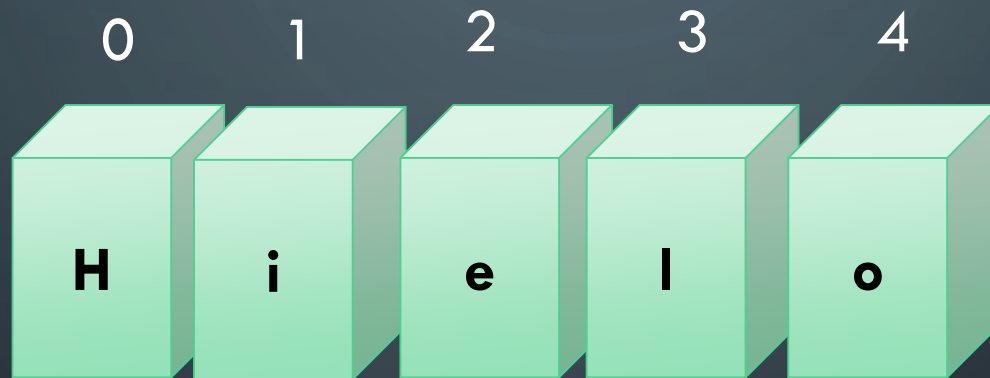
- Tipo `char[]`
- Tipo `string`

CADENAS

- Que es una cadena
- El tipo char
 - `cin.getline(var,longitud)`
- El tipo string
 - `getline(cin,var)`

CADENA

- Se denomina cadena a los datos que combinan números, letras y otros símbolos.



COMO INTRODUCIR CADENAS

- Tipo char

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

```
int main()
{
    char frase[20], palabra[20];
    cout << "Ingrese PALABRA " << endl;
    cin >> palabra;
    cout << "La palabra Ingresada es: " << palabra << endl;

    return 0;
}
```

CIN.GETLINE()

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

```
int main()
{
    char frase[20];
    cout<<"Ingrese una linea de texto"<<endl;
    cin.getline(frase,20);
    cout<<"la frase ingresada es: " << frase<<endl;

    return 0;
}
```

```
#include <iostream>
#include <string.h>
using namespace std;
int main()
{
    char cadena[20];
    cout << "Ingrese una cadena: \n";
    cin.getline(cadena,20);
    cout<<strlen(cadena);
    cin.get();
    for(int i=0;i<=strlen(cadena);i++)
    {
        if (cadena[i]>='A' && cadena[i]<='Z')
            cadena[i]+=32;
        else
        {
            if(cadena[i]=='Ñ')
                cadena[i]--;
        }
    }
    cout << cadena;
    return 0;
}
```

EJEMPLO

BIBLIOTECA DE MANEJO DE CADENAS

- La biblioteca `<string.h>` contiene un conjunto de funciones para manipular cadenas, cambiar caracteres, comparar cadenas, etc.
- Las funciones más elementales son:
- `strcpy(c1,c2);` //copia c2 en c1
- `strcat(c1,c2);` //añade c2 al final de c1
- `int strlen(cadena);` //devuelve la longitud de la cadena
- `int strcmp(c1,c2);` // devuelve cero si c1 es igual a c2

EJEMPLO

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

int main()
{
    char completo[80];
    char nombre[32]="Pedro";
    char apellidos[32]="Medario Arenas";

    //Construyendo el nombre completo --
    strcpy(completo,nombre); // completo <- "Pedro"
    strcat(completo, " "); //completo <- "Pedro "
    strcat(completo, apellidos); //completo <- "Pedro
Medario Arenas

    cout << "El nombre completo es: " << completo;
    return 0;
}
```

C++ STRING CLASS

- **¿Qué es la clase string?**

Es la encargada de modelar una cadena de caracteres, encapsulando mucha de la programación a bajo nivel y brindando métodos capaces de cumplir con muchas de las funciones que necesitaremos para manipular el tipo de datos más utilizado en cualquier programa.

- **¿Dónde la encontramos?**

La clase `string` está definida en la cabecera `string` (`#include <string>`) que forma parte del C++ estándar. Tan solo incluyéndola en nuestros módulos podremos acceder a ella. Pero si buscamos un poco más de comodidad, sea porque usaremos esa clase muchas veces en el código o por legibilidad, deberemos especificar que se utilizará el espacio de nombres estándar (`using namespace std;`).

EL TIPO STRING

- Es una clase de c++
 - `getline`
- Para su utilización no es necesario especificar la longitud en la definición, dado que se puede modificar de forma dinámica con la asignación de una nueva cadena
- `string cadena="Progral";`

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string frase, palabra[20];
    string frase1="Hola Mundo";
    string frase2(frase1);
    frase1="hola mundo";

    cout<< "ingrese una linea de texto \n";
    getline(cin,frase);

    cout << "La frase ingresada es: " << frase << endl;
    cout << "La frase1 es: " << frase1 << endl;
    cout << "La frase2 es: " << frase2 << endl;

    return 0;
}
```

EJEMPLO

- Elaborar un programa que lea una cadena y cuente el número de vocales que contiene.

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    int z,a,vocales=0;
    string frase;
    cout << "Ingrese una linea de texto"<<endl;
    getline(cin,frase);
    a=0;
    z=int(frase.length());
    while (a<=z)
    {
        if(frase[a]=='a')
            vocales++;
        a++;
    }
    cout<<vocales;

    return 0;
}
```

EJEMPLO2

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string texto1, texto2 = "Hola ", texto3("Que tal");
    texto1 = texto2 + texto3 + " estas? ";
    cout << texto1 << "\n";
    string subcadena (texto1, 2, 6); // 6 letras de texto1, desde la
tercera
    cout << subcadena << "\n";
    string subcadena2;
    subcadena2 = texto1.substr(0, 5); // 5 letras de texto1, desde el
comienzo
    texto1.insert(5, "Juan "); // Inserto un texto en la posicion 6
    cout << texto1 << "\n";
    texto2.replace(1, 2, "ad"); // Cambio 2 letras en la posicion 2
    cout << texto2 << "\n";
    cout << "La longitud de texto1 es " << texto1.size () << "\n";
    cout << "La tercera letra de texto1 es " << texto1[2] << " o bien " <<
texto1.at(2) << "\n";
    if (texto2 == "Hada ")
        cout << "Texto 2 es Hada\n";

    return 0;
}
```

EXPLICACIÓN

Casi todo se explica por sí solo. Aun así, vamos a dar un repaso rápido:

- Se puede crear una cadena sin valor inicial haciendo `string texto1;`
- Se le puede dar un valor inicial a la vez que se declara, haciendo `string texto2 = "Hola ";` o bien `string texto3("Que tal");`
- Se puede crear una cadena formada por varias, concateándolas (sumándolas), usando el signo `+`, así: `texto1 = texto2 + texto3 + " estas? ";`
- Se puede crear una subcadena a partir de un trozo de otra, la vez que se declara, así: `string subcadena (texto1, 2, 6);`
- O bien se puede extraer un fragmento posteriormente: `texto1.substr(0, 5);`
- Se puede insertar texto en el interior de una cadena: `texto1.insert(5, "Juan ");`
- O reemplazar ciertas letras por otras: `texto2.replace(1, 2, "ad");`
- Se puede saber el tamaño (cantidad de letras) de la cadena: `texto1.size()`
- Se puede acceder a una posición siguiendo el estándar de C: `texto1[2]`
- O bien usando la función "at": `texto1.at(2)`
- Se puede comprobar el valor de una cadena (el texto almacenado) con `==`, así: `if(texto2 == "Hada ") ...`

INVESTIGAR

- Esto es lo básico de las cadenas de texto en C++. Hay más (por ejemplo, un método "find" para ver si pertenece una cierta subcadena), pero eso queda propuesto como "ejercicio de investigación" para quien quiera profundizar más

EJERCICIOS

1. Escribir una función similar a `strlen` que pueda manejar cadenas sin terminador. Tip: se necesitará conocer y pasar la longitud de la cadena .
2. Escribir una función similar a `strcat` , `strcpy` y `strcmp` para que pueda manejar cadenas
3. Escribir una función que regrese verdad, si una cadena de entrada es un palíndromo. Un palíndromo es una palabra que se lee igual de izquierda a derecha, o de derecha a izquierda. Por ejemplo, ANA.
4. Escribir una función que convierta todos los caracteres de una cadena a mayúsculas.
5. Diseñar una función que cuenta cuantas palabras hay en una oración

CONT. EJERCICIOS

Buscando palabras.

Dada su experiencia ya con el manejo de las cadenas, demostrada en los ejercicios anteriores, en este ejercicio debe hacer un programa que pida al usuario una frase y también que pida una palabra para buscar dentro de dicha frase. El programa debe decirle en qué posición ha encontrado dicha frase, en caso de haberla encontrado, sino deberá mostrar un mensaje diciendo que no ha encontrado dicha palabra.

Por ejemplo:

Frase: Esta es una demostración del uso de las cadenas con el lenguaje C++

Palabra a buscar: uso

Resultado: Encontrada en la posición 30.