PRACTICA N° 2

OBJETIVOS.

Aprender los conceptos básicos del uso de COLAS, como estructuras de datos dinámicas.

BASE TEÓRICA.

La Clase *String* y algunos de sus Métodos más usados en C++

Una alternativa en el manejo de texto en ***C++*** es el "tipo ***string***" o la clase ***string*** la cual nos permite trabajar de forma más cómoda, de esta forma se pueden manipular dinámicamente las cadenas de texto definidas como un arreglo estático.

Los métodos más usados del tipo String.

Declaración y Asignación a un tipo string.

A un objeto declarado como tipo ***string***, se le pueden asignar otros objetos del mismo tipo, cadenas entre comillas dobles e incluso un único carácter a un tipo ***string***.

```
void  main()
{string cad_1("Hola mundo");
 string cad_2 = "Segunda forma";
 cad_2 = cad_1;  cad_1 = 'P';
}
```

Acceso a un carácter del tipo string.

Un elemento de la cadena se puede referenciar de la misma manera que se hace en un arreglo, con un índice entre corchetes **`cad_1[indice].`**

```
string cad_1 = "programa";
cout<<cad1_1[4];
// muestra el caracter 'r';
```

Comparaciones entre strings.

La comparación entre objetos ***string*** se hace directamente mediante el uso de los operadores **==, <=, >=, <, >, !=**.

```
string password;
getline(cin, password, '\n');
if(password == "xyzzy")
{  cout<<"Acceso permitido";}
```

Concatenación de Strings.

El operador **'+'** (más) permite concatenar dos o mas cadenas, como se muestra en el ejemplo.

Usando el operador +=, entonces se tiene C1+=C2 y con esto, la cadena 1 debería contener "**`Errar es de humanos`**"

```
string C1 = "Errar es ";
string C2 = "de humanos ";
string C3 = C1 + C2;
cout<<C3<<endl;
// muestra "Errar es de humanos"
C1 +=C2
```

Búsqueda de subcadenas o caracteres dentro del tipo string.

Se pueden hacer búsquedas de subcadenas dentro del objeto ***string***. En la cadena "**`Errar es de humanos`**" se puede buscar/obtener la posición donde está la palabra **`"de"`** usando **`cadena.find("de").`** Valor devuelto = 9, contando desde 0, o NULL, si no la encuentra.

```
string cadena = "Errar es de humanos";
if(cadena.find("de")==NULL)
   cout<<"cadena no existe"<<endl;
else
   cout<<"cadena si existe"<<endl;
```

Leer cadenas desde el teclado.

C++ dispone de dos métodos para leer texto desde el teclado: ***cin*** y ***getline()***, la diferencia entre ellos s está en que ***cin*** lee una cadena sin espacios, si por ejemplo se ingresa la cadena "Ciudad Guayana" ***cin*** retorna solo "Ciudad", por lo tanto ***cin***, se usa cuando es seguro que el dato es una palabra (no tiene 'espacios'). El método ***getline****(cin,Objeto_string)* recibe dos parámetros uno de ellos es `***cin***` que indica

que la lectura se hará desde la entrada estándar (teclado) y el siguiente parámetro es el objeto ***string*** donde se quiere almacenar la información ingresada.

```
string lectura1, lectura2;
cout<<"Ingrese la palabra: "; cin>>lectura1;
cin.ignore(256,'\n'); // limpia el buffer de entrada
getline(cin,lectura2);
cout<<"Lectura con cin>>lectura1 se obtuvo:     "         "<<lectura1<<endl;
cout<<"Lectura con getline(cin,lectura2) se obtuvo: "<<lectura2<<endl;
```

El método .swap()

swap recibe un parámetro de tipo ***string (C2)***, e intercambia su contenido con la cadena que llama el método (***C1.swap(C2)***),
ej: si C1 =”Ciudad” y C2=”Guayana” al hacer C1.***swap***(C2), el resultado será:
C1=”Guayana” y C2=”Ciudad”:
	`string C1="Ciudad"; string C2="Guayana";`
	`cout<<"\nC1 = "<<C1 <<endl;`
	`cout<<"str2 = "<<str2<<endl;`
	`str1.swap(str2);`
	`cout<<despues del intercambio.."<<endl;`
	`cout<<"C1 = "<<C1<<endl;`
	`cout<<"C2 = "<<C2<<endl;`

LONGITUD DE UN string.

Para determinar la longitud de un **string** se usan las funciones **length** o **size** las cuales retornan el número de caracteres del **string**:	`string mi_cadena = "diez caracteres.";`
	`int longitud = mi_cadena.length();`
	`// o int longitud = mi_cadena.size();`

Al igual que los elementos de un arreglo de caracteres los elementos de un **string** pueden ser indexados numéricamente.	`int i;`
	`for(i = 0; i < mi_cadena.length(); i++)`
	`{ cout<<mi_cadena[i]; }`

- **Funciones para manejo de Colas**

```
void PushCola(ptNODO &Cola, DATO Elem)
{   ptNODO A=Cola, N;
    NUEVO=new(NODO);
    NUEVO->Info=Elem; NUEVO->Sig=NULL;
    if(Cola==NULL)  Cola=NUEVO;
    else
    {   while(A->Sig!=NULL) A=A->Sig;
        A->Sig=N;
    }
}
```

```
int PopCola(ptNODO &Cola, DATO &Elem)
{   ptNODO P=Cola;
    if(P!=NULL)
    {   Cola=Cola->Sig;  Elem=P->Info;
        delete(P);
        return 1;
    }
    else return 0;
}
```

```
int FrenteCola(ptNODO  frente, DATO &Elem)
{   if(frente!=NULL)
    {   Elem = frente->Info;
        return 1;
    }
    else return 0;
}
```

```
void BorrarColaA(ptNODO &Cola)
{   int Elem;
    while(Cola!=NULL) PopCola(Cola,Elem);
}
```

```
void MostrarCola (ptNODO Cola)
{  ptNODO A=Cola;
   if (A != NULL)
   {  while(A!= NULL)
      {  cout<<A->Info<<endl;
           A=A->Sig;
      }
   }
   else cout<<"COLA Vacia";
      getch();
}
```

Desarrollo de la práctica.

1. Usando los datos de la práctica anterior escriba un programa que organice en cinco colas (Lunes, Martes, Miércoles, Jueves, Viernes) los datos referidos. Los datos se organizan según el terminal de la cedula, así las cedulas terminadas en 0 y1 van a la cola del Lunes, las terminadas en 2/3 van a la cola del Martes, 4/ 5 el Miércoles, 6/ 7 el Jueves y 8/9 el Viernes.

   [1] Cargar la tabla de datos.
   [2] Mostrar una cola específica.
   [3] Vaciar arreglo, mostrando datos que salen.
   [4] Opción Extra.