

## Práctica 01

DOCENTE	CARRERA	CURSO
MSc. Vicente Enrique Machaca Arceda	Escuela Profesional de Ingeniería de Software	Fundamentos de Lenguajes de Programación

PRÁCTICA	TEMA	DURACIÓN
01	Paradigmas de programación	3 horas

### 1. Datos de los estudiantes

- Grupo: Equipo alfa buena maravilla onda dinamita escuadrón lobo
- Integrantes:
  - Andree Alvarez Guzman
  - Jonathan Aguirre Soto

### 2. Ejercicios

1. Utilice la programación imperativa para implementar un algoritmo. Usted decide que algoritmo implementar. Describa el algoritmo y despues implementelo en el lenguaje de su preferencia.

- Solución: Código Fuente

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

struct Lista{
    int n;
    Lista *sgte;
}*cab = nullptr, *aux=nullptr, *aux1=nullptr;

int op=0;
int insertar(void){
    do{
        aux=new Lista;
        cout<<"Ingrese numero: ";
        cin>>aux->n;

        aux->sgte = nullptr;

        if(cab==nullptr){
            cab=aux;
        }
        else{
```

```
        aux1->sgte=aux;
    }
    aux1=aux;
    cout<<"Desea seguir ingresando?: 1: SI, 0: NO"<<endl;
    cin>>op;
} while (op==1);
return 1;
};

void mostrar_lista(void){
    int n=0;
    if (cab!=NULL){
        for (aux=cab; aux!=nullptr; aux=aux->sgte){
            if (n>0){
                cout<<"->";
            }
            cout<<aux->n;
            n=1;
        }
    }
    else{
        cout<<"La Lista esta vacia";
    }
    cout<<endl;
}

int buscar_lista(){
    int v=0;
    cout<<"Ingresar nodo a buscar: ";
    cin>>v;
    if (cab!=nullptr){
        for (aux=cab; aux!=nullptr; aux=aux->sgte){
            if (aux->n==v){
                cout<<"El elemento si existe"<<endl;
                return 1;
            }
        }
        cout<<"El elemento no existe"<<endl;
        return 0;
    }
    else{
        cout<<"La lista esta vacia"<<endl;
        return 0;
    }
};

Lista* consulta(int n){
    Lista* temp=new Lista();
    if (cab!=nullptr){
        for (aux=cab; aux!=nullptr; aux=aux->sgte){
            if (aux->n==n){
                temp=aux;
            }
        }
    }
};
```

```
        return temp;
    }
}
cout<<"El elemento no existe"<<endl;
}
return 0;
}

int eliminar_lista(){
    Lista *temp = new Lista();
    //temp = consulta(n);
    /*if (!temp){
        cout << "El registro no existe\n";
        return 1;
    }*/
    int n=0;
    cout<<"Ingrese nodo a eliminar: ";
    cin>>n;
    for(aux=cab; aux!=nullptr; aux=aux->sgte){
        if((aux->n==n) && (aux->sgte!=nullptr) && (aux==cab)){
            //cout<<"Elimina al inicio con siguiente"<<endl;
            //temp=aux;
            cab=aux->sgte;
            delete aux;
            return 1;
        }
        if((aux->n==n) && (aux->sgte==nullptr) && (aux==cab)){
            //cout<<"Elimina al inicio unico"<<endl;
            cab=nullptr;
            return 1;
        }
        else if((aux->n==n) && (aux->sgte==nullptr) && (aux!=cab)){
            //cout<<"Elimina al final"<<endl;
            temp->sgte=nullptr;
            delete aux;
            return 1;
        }
        else if((aux->n==n) && (aux->sgte!=nullptr) && (aux!=cab)){
            //cout<<"Elimina al medio"<<endl;
            //temp=aux;
            temp->sgte=aux->sgte;
            delete aux;
            return 1;
        }
        temp=aux;
    }
    cout<<"El registro no existe"<<endl;
    return 0;
}

int main(){
    int op=0;
```

```
while (op!=5){
    cout<<"Menu:-----"<<endl;
    cout<<"1. Insertar"<<endl;
    cout<<"2. Mostrar"<<endl;
    cout<<"3. Buscar"<<endl;
    cout<<"4. Eliminar"<<endl;
    cout<<"5. Salir"<<endl;
    cin>>op;
    switch (op){
        case 1:
            insertar ();
            break;
        case 2:
            mostrar_lista ();
            break;
        case 3:
            buscar_lista ();
            break;
        case 4:
            eliminar_lista ();
            break;
        case 5:
            break;
        default:
            cout<<"No existe la opcion"<<endl;
            break;
    }
}
```

- Capturas de la ejecución:

```
> clang++-7 -pthread -std=c++17 -o main main.cpp
> ./main
Menu:-----
1. Insertar
2. Mostrar
3. Buscar
4. Eliminar
5. Salir
█
```

- Insertar.

```
> clang++-7 -pthread -std=c++17 -o main main.cpp
> ./main
Menu:-----
1. Insertar
2. Mostrar
3. Buscar
4. Eliminar
5. Salir
1
Ingrese numero: 1
Desea seguir ingresando?: 1: SI, 0: NO
1
Ingrese numero: 2
Desea seguir ingresando?: 1: SI, 0: NO
0
Menu:-----
1. Insertar
2. Mostrar
3. Buscar
4. Eliminar
5. Salir
█
```

- Mostrar.

```
> clang++-7 -pthread -std=c++17 -o main main.cpp
> ./main
Menu:-----
1. Insertar
2. Mostrar
3. Buscar
4. Eliminar
5. Salir
1
Ingrese numero: 1
Desea seguir ingresando?: 1: SI, 0: NO
1
Ingrese numero: 2
Desea seguir ingresando?: 1: SI, 0: NO
0
Menu:-----
1. Insertar
2. Mostrar
3. Buscar
4. Eliminar
5. Salir
2
1->2
Menu:-----
1. Insertar
2. Mostrar
3. Buscar
4. Eliminar
5. Salir
█
```

- Eliminar.

```
3 clang++-7 -pthread -std=c++17 -o main main.cpp
3 ./main
Menu:-----
1. Insertar
2. Mostrar
3. Buscar
4. Eliminar
5. Salir
1
Ingrese numero: 1
Desea seguir ingresando?: 1: SI, 0: NO
1
Ingrese numero: 2
Desea seguir ingresando?: 1: SI, 0: NO
0
Menu:-----
1. Insertar
2. Mostrar
3. Buscar
4. Eliminar
5. Salir
2
1->2
Menu:-----
1. Insertar
2. Mostrar
3. Buscar
4. Eliminar
5. Salir
3
Ingresar nodo a buscar: 3
El elemento no existe
Menu:-----
1. Insertar
2. Mostrar
3. Buscar
4. Eliminar
5. Salir
```

- Otras operaciones.

```
Menu:-----
1. Insertar
2. Mostrar
3. Buscar
4. Eliminar
5. Salir
2
1->2
Menu:-----
1. Insertar
2. Mostrar
3. Buscar
4. Eliminar
5. Salir
3
Ingresar nodo a buscar: 3
El elemento no existe
Menu:-----
1. Insertar
2. Mostrar
3. Buscar
4. Eliminar
5. Salir
4
Ingrese nodo a eliminar: 2
Menu:-----
1. Insertar
2. Mostrar
3. Buscar
4. Eliminar
5. Salir
2
1
Menu:-----
1. Insertar
2. Mostrar
3. Buscar
4. Eliminar
5. Salir
```



- Salir.

```
Menu:-----
1. Insertar
2. Mostrar
3. Buscar
4. Eliminar
5. Salir
2
1->2
Menu:-----
1. Insertar
2. Mostrar
3. Buscar
4. Eliminar
5. Salir
3
Ingresar nodo a buscar: 3
El elemento no existe
Menu:-----
1. Insertar
2. Mostrar
3. Buscar
4. Eliminar
5. Salir
4
Ingrese nodo a eliminar: 2
Menu:-----
1. Insertar
2. Mostrar
3. Buscar
4. Eliminar
5. Salir
2
1
Menu:-----
1. Insertar
2. Mostrar
3. Buscar
4. Eliminar
5. Salir
5
>
```

2. Implemente un ejemplo de programación declarativa, puede utilizar cualquier lenguaje a excepción de SQL.

- Solución: Código Fuente

#### Ejemplo en PHP

Comencemos con esta lista participantes que nos servirá para crear una lista con nombres.

Con la programación imperativa (declaramos los pasos y decidimos lo que queremos haga):

```
$listaparticipantes = [1 => 'Peter', 2 => 'Hans', 3 => 'Sarah'];  
$nombres = [];  
foreach ($listaparticipantes as $id => $apellido)  
{  
    $nombres[] = $apellido;  
}
```

Con la programación declarativa (declaramos los resultados pero no el paso a paso):

```
$nombres = array_values($listaparticipantes);
```

#### Ejemplo en Python

Programación imperativa:

```
small_nums = []  
for i in range(20):  
    if i < 5:  
        small_nums.append(i)
```

Programación declarativa:

```
small_nums = [x for x in range(20) if x < 5]
```

- Capturas de la ejecución:

- Forma imperativa PHP.

```
1  <?php
2  // Your code here!
3
4  $listaparticipantes = [1 => 'Peter', 2 => 'Hans', 3 => 'Sarah'];
5
6  $listaparticipantes = [1 => 'Peter', 2 => 'Hans', 3 => 'Sarah'];
7  $nombres = [];
8  foreach ($listaparticipantes as $id => $apellido) {
9      $nombres[] = $apellido;
10 }
11
12 print_r($nombres);
13
14 ?>
15
```

Ejecutar (Ctrl-Enter)

Salida Entrada Comments 0

```
Array
(
    [0] => Peter
    [1] => Hans
    [2] => Sarah
)
```

- Forma declarativa PHP.

```
1 <?php
2 // Your code here!
3
4 $listaparticipantes = [1 => 'Peter', 2 => 'Hans', 3 => 'Sarah'];
5 $nombres = array_values($listaparticipantes);
6
7
8 print_r($nombres);
9
10 ?>
11
```



Ejecutar (Ctrl-Enter)


Salida Entrada Comments 0

```
Array
(
    [0] => Peter
    [1] => Hans
    [2] => Sarah
)
```

- Forma imperativa Python.

```
1  # coding: utf-8
2  # Your code here!
3
4  small_nums = []
5  for i in range(20):
6      if i < 5:
7          small_nums.append(i)
8
9  print(small_nums)
10
```

 Ejecutar (Ctrl-Enter) 

Salida Entrada Comments 

[0, 1, 2, 3, 4]

- Forma declarativa Python.

```
1 # coding: utf-8
2 # Your code here!
3
4 small_nums = [x for x in range(20) if x < 5]
5 .....
6 print(small_nums)
```

Ejecutar (Ctrl-Enter)

Salida Entrada Comments 0

[0, 1, 2, 3, 4]

3. Enlace de Github:

<https://github.com/Jona2010/Fundamentos-de-Lenguaje-de-la-Programaci-n>