



Universidad Tres Culturas



PLANTEL “LONDRES”

Ingeniería en Sistemas Computacionales

Reporte de Prácticas
Del 22 al 26 de julio.

Presenta:

García Salas Natalia

Grupo: “A”

Turno: Matutino

Docente: José Guadalupe Sánchez
Hernández

Asignatura: Estructura de Datos

Fecha de entrega: 27 de julio de 2024.

Introducción	1
Desarrollo	1
1. Práctica Pilas	1
Diagrama de Flujo	1
Código	3
Salida de escritorio	6
Documentación por bloques de código	7
2. Examen	7
Diagrama de flujo	7
Código	10
Salida de Escritorio	14
Documentación por bloques	14

Introducción

Como estudiante de Ingeniería en Sistemas Computacionales es importante poner en práctica los conocimientos teóricos aprendidos en el aula de clases. Es por eso que se realizarán los siguientes ejercicios, para reforzar los conocimientos adquiridos en niveles anteriores de la carrera.

Desarrollo

1. Práctica Pilas

Diagrama de Flujo

Inicio

|

v

Inicializar pila.cima = -1

|

v

system("cls")

|

v

eleccion = intOpcion()

|

v

+-----+

| eleccion != 4 |

| /\ |

| | |

```

|   v   |
| (switch eleccion) |
|   |   |
|   v   |
| eleccion == 1? -----+-----> LlenarPila(&pila)
|   |   |
|   v   |
| eleccion == 2? -----+-----> MostrarPila(&pila)
|   |   |
|   v   |
| eleccion == 3? -----+-----> EliminarPila(&pila)
|   |   |
|   v   |
| else -----+-----> printf("Opción incorrecta\n")
+-----+
|
v
FIN

```

LlenarPila(p):

```

|
v
(p->cima < 100 - 1) ?
|
+-----+
|           |
v           v
TRUE        FALSE
|           |
v           v
printf("Ingresa el dato:\n");
scanf("%d", &dato);
p->cima++;
p->dato[p->cima] = dato;
|
v
RETURN

```

MostrarPila(p):

```

|
v
(p->cima == -1) ?
|
+-----+
|           |
v           v
TRUE        FALSE

```

```

|
v
printf("La pila está vacía.\n");
printf("Contenido de la pila:\n");
for (i = p->cima; i >= 0; i--) {
    printf("%d\n", p->dato[i]);
}

```

```

|
v
RETURN

```

EliminarPila(p):

```

|
v
(p->cima == -1) ?
|
+-----+
|         |
v         v
TRUE      FALSE
|         |
v         v
printf("La pila está vacía.\n");
printf("Último elemento eliminado.\n");
|
v
RETURN

```

intOpcion():

```

|
v
printf("\nSeleccione una opción:\n");
printf("1. Nuevo nodo\n");
printf("2. Mostrar pila\n");
printf("3. Eliminar pila\n");
printf("4. Salir\n");
scanf("%d", &opcion);
|
v
RETURN opcion

```

Código

//Programa hecho por @Natalia Garcia
//24/07/24

```
#include <stdio.h>
#include <stdlib.h>
```

```
struct Pila {
    int dato[100];
    int cima;
};
```

```
// Prototipos de funciones
int intOpcion(void);
void LlenarPila(struct Pila *p);
void MostrarPila(struct Pila *p);
void EliminarPila(struct Pila *p);
```

```
int main() {
    struct Pila pila;
    int eleccion;
    pila.cima = -1;
    system("cls");
    do{
        switch(eleccion){
            case 1: //Agregar elementos
                LlenarPila(&pila);
                break;
            case 2: //Mostrar pila
                MostrarPila(&pila);
                break;
            case 3: //Eliminar último elemento
                EliminarPila(&pila);
                break;
            default:
                if (eleccion < 1 || eleccion > 6) {
                    printf("Opción incorrecta\n");
                }
                break;
        }
    }while((eleccion = intOpcion()) != 4);
    return 0;
}
```

```
void LlenarPila(struct Pila *p) {
    int dato;
    if (p->cima < 100 - 1) {
        printf("Ingresa el dato:\n");
        scanf("%d",&dato);
        p->cima++;
        p->dato[p->cima] = dato;
    }
```

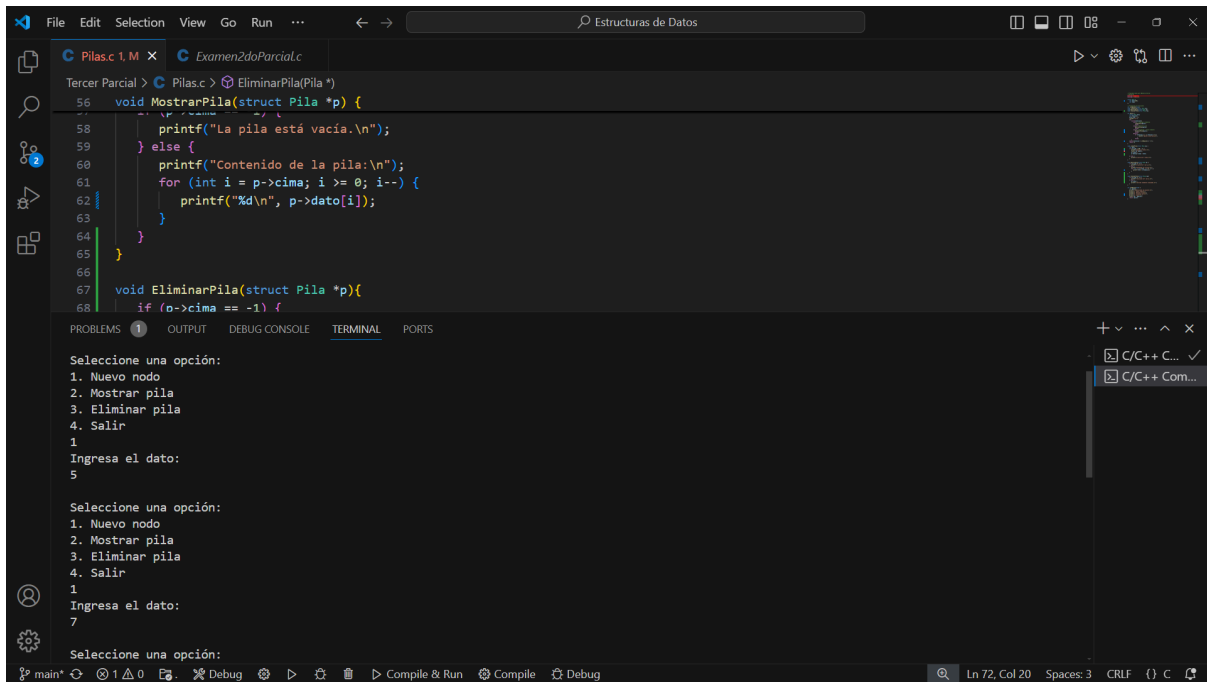
```
} else {
    printf("La pila está llena.\n");
}
}

void MostrarPila(struct Pila *p) {
    if (p->cima == -1) {
        printf("La pila está vacía.\n");
    } else {
        printf("Contenido de la pila:\n");
        for (int i = p->cima; i >= 0; i--) {
            printf("%d\n", p->dato[i]);
        }
    }
}

void EliminarPila(struct Pila *p){
    if (p->cima == -1) {
        printf("\nLa pila está vacía.\n");
    } else {
        p->cima--;
        printf("\nÚltimo elemento eliminado.\n");
    }
}

int intOpcion(void) {
    int opcion;
    printf("\nSelecione una opción:\n");
    printf("1. Nuevo nodo\n");
    printf("2. Mostrar pila\n");
    printf("3. Eliminar pila\n");
    printf("4. Salir\n");
    scanf("%d", &opcion);
    return opcion;
}
```

Salida de escritorio

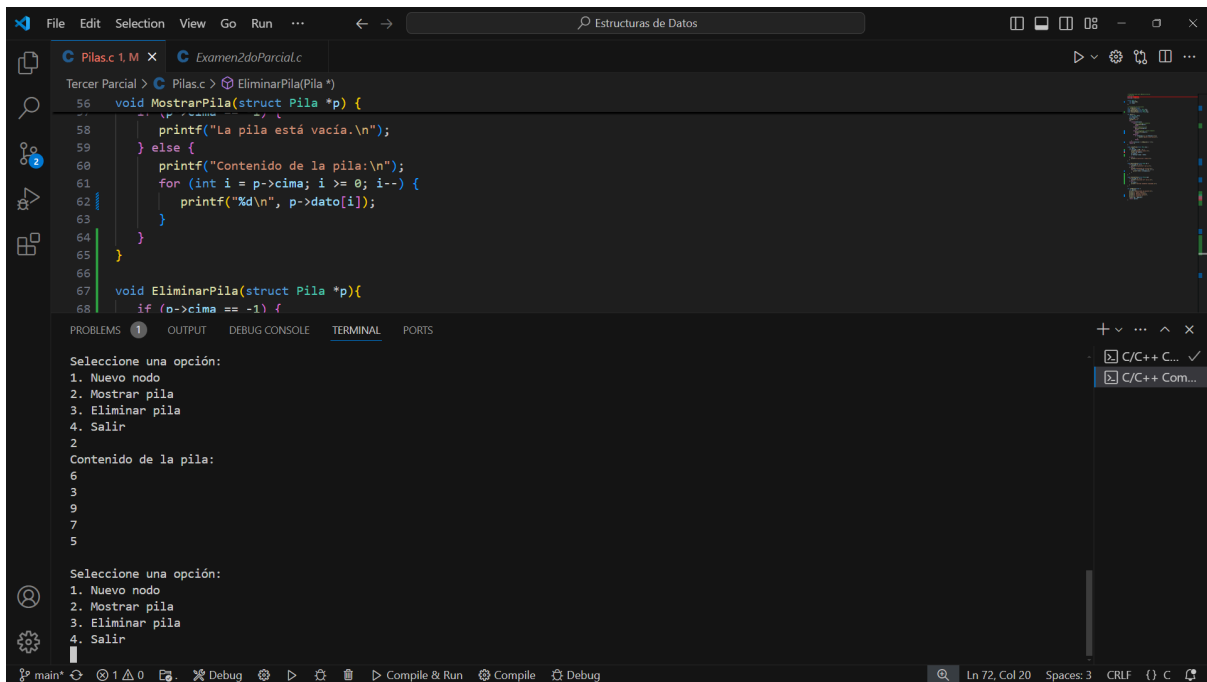


```
File Edit Selection View Go Run ... Estructuras de Datos
Pilas.c 1, M x Examen2doParcial.c
Tercer Parcial > C Pilas.c > EliminarPila(Pila *)
56 void MostrarPila(struct Pila *p) {
57     printf("La pila está vacía.\n");
58 } else {
59     printf("Contenido de la pila:\n");
60     for (int i = p->cima; i >= 0; i--) {
61         printf("%d\n", p->dato[i]);
62     }
63 }
64 }
65 }
66 }
67 void EliminarPila(struct Pila *p){
68     if (p->cima == -1) {
69         printf("La pila está vacía.\n");
70     } else {
71         printf("Contenido de la pila:\n");
72         for (int i = p->cima; i >= 0; i--) {
73             printf("%d\n", p->dato[i]);
74         }
75     }
76 }
77 }
78 }
79 }
80 }
81 }
82 }
83 }
84 }
85 }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS
Seleccione una opción:
1. Nuevo nodo
2. Mostrar pila
3. Eliminar pila
4. Salir
1
Ingresa el dato:
5

Seleccione una opción:
1. Nuevo nodo
2. Mostrar pila
3. Eliminar pila
4. Salir
1
Ingresa el dato:
7

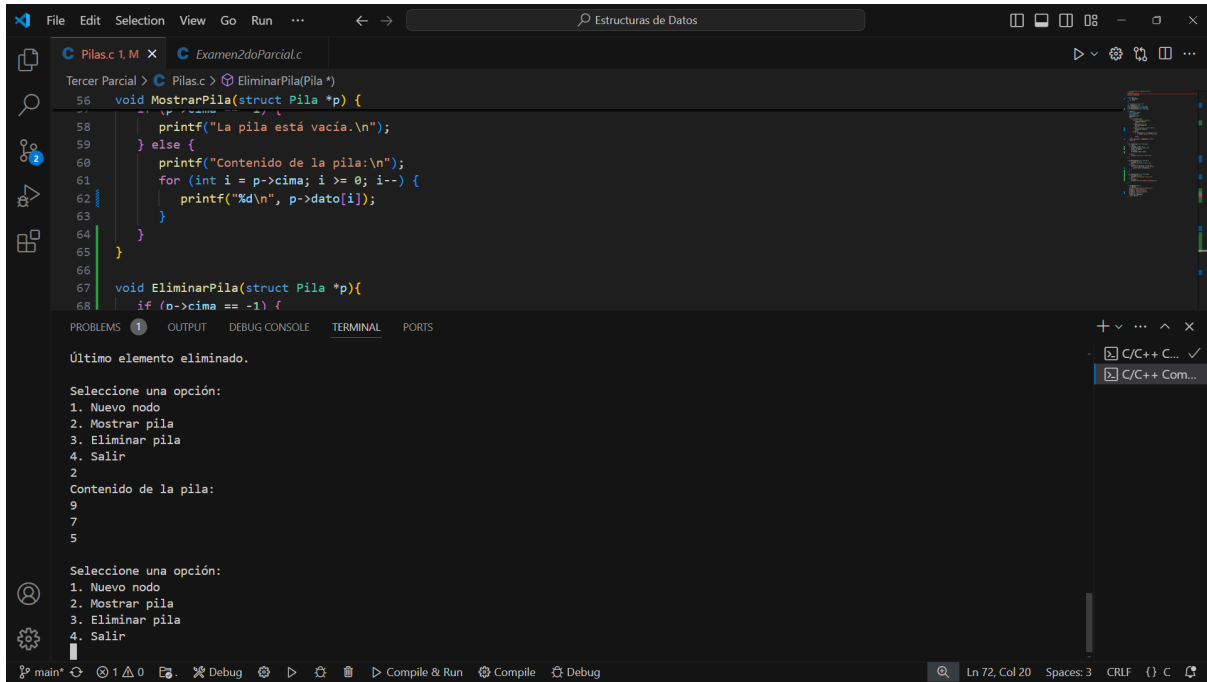
Seleccione una opción:
```



```
File Edit Selection View Go Run ... Estructuras de Datos
Pilas.c 1, M x Examen2doParcial.c
Tercer Parcial > C Pilas.c > EliminarPila(Pila *)
56 void MostrarPila(struct Pila *p) {
57     printf("La pila está vacía.\n");
58 } else {
59     printf("Contenido de la pila:\n");
60     for (int i = p->cima; i >= 0; i--) {
61         printf("%d\n", p->dato[i]);
62     }
63 }
64 }
65 }
66 }
67 void EliminarPila(struct Pila *p){
68     if (p->cima == -1) {
69         printf("La pila está vacía.\n");
70     } else {
71         printf("Contenido de la pila:\n");
72         for (int i = p->cima; i >= 0; i--) {
73             printf("%d\n", p->dato[i]);
74         }
75     }
76 }
77 }
78 }
79 }
80 }
81 }
82 }
83 }
84 }
85 }
86 }
87 }
88 }
89 }
90 }
91 }
92 }
93 }
94 }
95 }
96 }
97 }
98 }
99 }
100 }

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS
Seleccione una opción:
1. Nuevo nodo
2. Mostrar pila
3. Eliminar pila
4. Salir
2
Contenido de la pila:
6
3
9
7
5

Seleccione una opción:
1. Nuevo nodo
2. Mostrar pila
3. Eliminar pila
4. Salir
1
```



```
Tercer Parcial > C Pilas.c > EliminarPila(Pila *)
56 void MostrarPila(struct Pila *p) {
57     if (p->cima == -1) {
58         printf("La pila está vacía.\n");
59     } else {
60         printf("Contenido de la pila:\n");
61         for (int i = p->cima; i >= 0; i--) {
62             printf("%d\n", p->dato[i]);
63         }
64     }
65 }
66
67 void EliminarPila(struct Pila *p){
68     if (p->cima == -1) {
69         printf("La pila está vacía.\n");
70     } else {
71         p->cima--;
72     }
73 }
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

Último elemento eliminado.

Seleccione una opción:

1. Nuevo nodo
2. Mostrar pila
3. Eliminar pila
4. Salir

2

Contenido de la pila:

9
7
5

Seleccione una opción:

1. Nuevo nodo
2. Mostrar pila
3. Eliminar pila
4. Salir

1

Documentación por bloques de código

- Bloque Verde

Declaración de variables

- Bloque Rojo

Lectura de datos

- Bloque Azul

Operaciones

- Bloque Naranja

Resultado final

2. Examen

Diagrama de flujo

Inicio

```

|
v
ptrInicio = NULL
|
v
ptrNuevo = malloc(sizeof(NodoCalificacion))
|
v
strcpy(ptrNuevo->apellido, "Perez")
|
v
ptrNuevo->calificacion = 91.5
|
v
ptrInicio = ptrNuevo
|
v
ptrAnterior = NULL
|
v
ptrActual = ptrInicio
|
v
+-----+
| (ptrActual != NULL && |
| strcmp(ptrActual->apellido, "Fernandez") < 0) |
| /\ |
| | |
| v |
| TRUE FALSE |
| | |
| v v |
| ptrAnterior = ptrActual ptrNuevo = malloc(sizeof(NodoCalificacion)) |
| ptrActual = ptrActual->ptrSiguiente strcpy(ptrNuevo->apellido, "Fernandez") |
| ptrNuevo->calificacion = 85.0 |
| ptrNuevo->ptrSiguiente = ptrActual |
| if (ptrAnterior == NULL) |
| | |
| v |
| ptrInicio = ptrNuevo |
| else |
| | |
| v |

```

```
| ptrAnterior->ptrSiguiente = ptrNuevo |
```

```
+-----+
```

```
|
```

```
v
```

```
ptrActual = ptrInicio
```

```
|
```

```
v
```

```
+-----+
```

```
| (ptrActual != NULL) |
```

```
| /\ |
```

```
| | |
```

```
| v |
```

```
| TRUE FALSE |
```

```
| | |
```

```
| v v |
```

```
| printf("Apellido: %s, FIN
```

```
| Calificación: %.1f\n",
```

```
| ptrActual->apellido,
```

```
| ptrActual->calificacion)
```

```
| ptrActual = ptrActual->ptrSiguiente
```

```
+-----+
```

```
|
```

```
v
```

```
ptrActual = ptrInicio
```

```
|
```

```
v
```

```
+-----+
```

```
| (ptrActual != NULL) |
```

```
| /\ |
```

```
| | |
```

```
| v |
```

```
| TRUE FALSE |
```

```
| | |
```

```
| v v |
```

```
| ptrTemp = ptrActual FIN
```

```
| ptrActual = ptrActual->ptrSiguiente
```

```
| free(ptrTemp)
```

```
+-----+
```

```
|
```

```
v
```

```
ptrInicio = NULL
```

```
|
```

```
v
```

```
RETURN
```

Código

//Programa hecho por @Natalia García
//25/07/24

```
#include <stdio.h>
#include <string.h>

struct nodoCalificacion {
    char apellido[20];
    double calificacion;
    struct nodoCalificacion *ptrSiguiente;
};

typedef struct nodoCalificacion NodoCalificacion;
typedef NodoCalificacion *ptrNodoCalificacion;

int main() {
    ptrNodoCalificacion ptrInicio = NULL;

    ptrNodoCalificacion ptrNuevo = (ptrNodoCalificacion)malloc(sizeof(NodoCalificacion));
    strcpy(ptrNuevo->apellido, "Perez");
    ptrNuevo->calificacion = 91.5;
    ptrInicio = ptrNuevo;

    ptrNodoCalificacion ptrAnterior = NULL, ptrActual = ptrInicio;
    while (ptrActual != NULL && strcmp(ptrActual->apellido, "Fernandez") < 0) {
        ptrAnterior = ptrActual;
        ptrActual = ptrActual->ptrSiguiente;
    }
    ptrNuevo = (ptrNodoCalificacion)malloc(sizeof(NodoCalificacion));
    strcpy(ptrNuevo->apellido, "Fernandez");
    ptrNuevo->calificacion = 85.0;
    ptrNuevo->ptrSiguiente = ptrActual;
    if (ptrAnterior == NULL) {
        ptrInicio = ptrNuevo;
    } else {
        ptrAnterior->ptrSiguiente = ptrNuevo;
    }

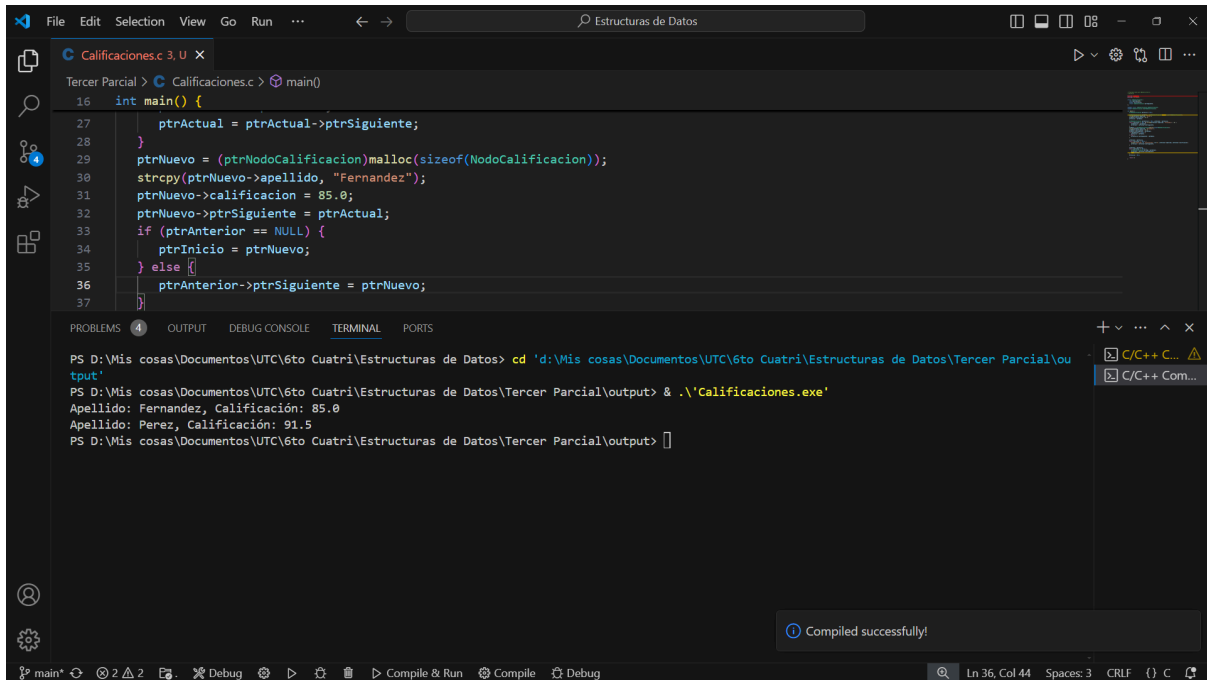
    ptrActual = ptrInicio;
    while (ptrActual != NULL) {
        printf("Apellido: %s, Calificación: %.1f\n", ptrActual->apellido, ptrActual->calificacion);
        ptrActual = ptrActual->ptrSiguiente;
    }

    ptrActual = ptrInicio;
    while (ptrActual != NULL) {
```

```
ptrNodoCalificacion ptrTemp = ptrActual;
ptrActual = ptrActual->ptrSiguiente;
free(ptrTemp);
}
ptrInicio = NULL;

return 0;
}
```

Salida de Escritorio



```
Calificaciones.c 3, U
Tercer Parcial > Calificaciones.c > main()
16 int main() {
27     ptrActual = ptrActual->ptrSiguiente;
28 }
29 ptrNuevo = (ptrNodoCalificacion)malloc(sizeof(NodoCalificacion));
30 strcpy(ptrNuevo->apellido, "Fernandez");
31 ptrNuevo->calificacion = 85.0;
32 ptrNuevo->ptrSiguiente = ptrActual;
33 if (ptrAnterior == NULL) {
34     ptrInicio = ptrNuevo;
35 } else {
36     ptrAnterior->ptrSiguiente = ptrNuevo;
37 }
```

```
PS D:\Mis cosas\Documentos\UTC\6to Cuatri\Estructuras de Datos\Tercer Parcial\ou
tput>
PS D:\Mis cosas\Documentos\UTC\6to Cuatri\Estructuras de Datos\Tercer Parcial\output> & .\Calificaciones.exe
Apellido: Fernandez, Calificación: 85.0
Apellido: Perez, Calificación: 91.5
PS D:\Mis cosas\Documentos\UTC\6to Cuatri\Estructuras de Datos\Tercer Parcial\output> 
```

Compiled successfully!

Documentación por bloques

● Bloque Verde

Declaración de variables

● Bloque Rojo

Lectura de datos

● Bloque Azul

Operaciones

● Bloque Naranja

Resultado final