Estructuras de Datos: Colas

Cosijopii García

Universidad del Istmo

8 de noviembre de 2024

Contenido

Introducción a C

- C es un lenguaje de programación de propósito general.
- Es eficiente y se utiliza ampliamente para programación de sistemas.
- Se introdujo por primera vez en 1972.
- Estructurado, de tipado estático y con manejo explícito de la memoria.

Tipos de Datos en C

- **int**: Enteros (e.g., 1, 100, -3)
- **float**: Números de punto flotante de precisión simple (e.g., 3.14, -0.001)
- **double**: Números de punto flotante de doble precisión
- **char**: Caracteres individuales (e.g., 'a', 'z', '$')
- **void**: Representa la ausencia de valor (e.g., funciones que no retornan valor)

Declaración y Definición de Funciones en C

Declaración de una función:

```
int suma(int a, int b);
```

Definición de una función:

```
int suma(int a, int b) {
    return a + b;
}
```

- Las funciones en C permiten modularizar el código.
- Se pueden reutilizar y organizar mejor el código.

Llamada de Funciones

Ejemplo de llamada a una función:

```
#include <stdio.h>

int suma(int a, int b) {
    return a + b;
}

int main() {
    int resultado = suma(4, 6);
    printf("El resultado es: %d\n", resultado);
    return 0;
}
```

- La función ‘suma‘ es llamada con los argumentos 4 y 6.
- El valor devuelto (10) se almacena en la variable ‘resultado‘.

Paso de Parámetros en C

- **Paso por valor**: Se pasa una copia del valor a la función.
- **Paso por referencia**: Se pasa la dirección de memoria del valor, permitiendo modificar el valor original.

Paso por Referencia en C I

Ejemplo de paso por referencia:

```
#include <stdio.h>

void incrementar(int *num) {
    (*num)++;  // Incrementa el valor apuntado
}

int main() {
    int x = 5;
    incrementar(&x);  // Pasa la direcci n de x
    printf("%d\n", x);  // Imprime 6
    return 0;
}
```

- ‘x‘ pasa la dirección de memoria de la variable ‘x‘.
- Dentro de la función ‘incrementar‘, se modifica el valor almacenado en esa dirección.

Retorno de Valores en C

- Una función en C puede retornar un valor utilizando la palabra clave **return**.
- El tipo del valor retornado debe coincidir con el tipo declarado de la función.

Ejemplo de función que retorna un valor:

```
int cuadrado(int num) {
    return num * num;
}

int main() {
    int resultado = cuadrado(7);
    printf("El cuadrado de 7 es: %d\n", resultado);
    return 0;
}
```

Estructuras en C

- Una **estructura** es un tipo de dato definido por el usuario que permite agrupar variables de diferentes tipos bajo un mismo nombre.
- Útil para representar entidades del mundo real, como registros o nodos.
- Se declara con la palabra clave `struct`.

Definición de una Estructura

Ejemplo: Definición de un estudiante

```
#include <stdio.h>

// Definici n de la estructura
struct Estudiante {
    char nombre[50];
    int matricula;
    float promedio;
};

int main() {
    struct Estudiante estudiante1;

    // Asignaci n de valores
    strcpy(estudiante1.nombre, "Juan Perez");
    estudiante1.matricula = 12345;
    estudiante1.promedio = 8.9;
```

Punteros y Estructuras

- Se pueden utilizar punteros para manipular estructuras.
- El operador -> se utiliza para acceder a miembros de una estructura a través de un puntero.

Ejemplo:

```
struct Estudiante est;
struct Estudiante *ptr;

ptr = &est;
ptr->matricula = 54321;
```

Punteros y Arreglos en C

- Un **arreglo** es una colección de datos del mismo tipo, almacenados en posiciones de memoria contiguas.
- Un **puntero** es una variable que almacena la dirección de memoria de otra variable.
- Un puntero puede apuntar al primer elemento de un arreglo, y se pueden recorrer los elementos usando el puntero.

Ejemplo: Punteros y Arreglos

Ejemplo: Recorrer un arreglo con punteros

```
#include <stdio.h>

int main() {
    int arr[] = {10, 20, 30, 40, 50};
    int *ptr = arr;  // ptr apunta al primer elemen

    for (int i = 0; i < 5; i++) {
        printf("Valor: %d\n", *(ptr + i));  // Desr
    }

    return 0;
}
```

- 'ptr' apunta al primer elemento del arreglo.
- Al incrementar 'ptr', accedemos a los siguientes elementos del arreglo.

Cadenas de Caracteres en C

- Una **cadena de caracteres** es un arreglo de tipo `char`.
- Se finaliza con el carácter nulo '`\0`'.
- Pueden manipularse usando funciones de la biblioteca estándar como `strlen`, `strcpy`, `strcat`, etc.

Ejemplo: Uso de Cadenas de Caracteres I

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

int main() {
    char nombre[50];
    char saludo[50] = "Hola, ";

    printf("Introduce tu nombre: ");
    scanf("%s", nombre);

    strcat(saludo, nombre);  // Concatena las caden
    printf("%s\n", saludo);

    return 0;
}
```

- `strcat` concatena dos cadenas.

Ejemplo: Uso de Cadenas de Caracteres II

- La cadena de destino debe tener suficiente espacio para almacenar el resultado.

Funciones Comunes con Cadenas de Caracteres

- `strlen`: Calcula la longitud de una cadena (sin contar el carácter nulo).
- `strcpy`: Copia una cadena a otra.
- `strcat`: Concatena dos cadenas.
- `strcmp`: Compara dos cadenas.

Ejemplo: Uso de strlen y strcpy I

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

int main() {
    char nombre[50] = "Ana";
    char copia[50];

    // Copiar cadena
    strcpy(copia, nombre);
    printf("Copia: %s\n", copia);

    // Calcular longitud de la cadena
    printf("Longitud del nombre: %lu\n", strlen(nom

    return 0;
}
```

Ejemplo: Uso de strlen y strcpy II

- `strcpy` copia la cadena `nombre` en `copia`.
- `strlen` devuelve la longitud de la cadena.

¿Qué es una Cola?

- Una **cola** es una estructura de datos que sigue la política **FIFO** (*First In, First Out*).
- Esto significa que el primer elemento en entrar es el primero en salir.
- Se utilizan dos punteros:
  - **Frente**: Indica el primer elemento de la cola.
  - **Final**: Indica el último elemento agregado a la cola.
- Ejemplos de aplicaciones incluyen:
  - Manejo de procesos en un sistema operativo.
  - Simulación de filas de espera (en bancos, supermercados, etc.).

Operaciones Básicas en una Cola

- **Encolar (enqueue)**: Inserta un nuevo elemento al final de la cola.
- **Desencolar (dequeue)**: Remueve el elemento en el frente de la cola.
- **Frente (front)**: Muestra el elemento en el frente sin eliminarlo.
- **isEmpty**: Verifica si la cola está vacía.
- **isFull**: Verifica si la cola está llena (si se implementa con un arreglo de tamaño fijo).

Cola Simple

- En una **cola simple**, los elementos se encolan por el final y se desencolan por el frente.
- Ejemplo de una cola de 5 elementos:

Frente Final

10	20	30	40	50

- **Encolar (enqueue)**: Se agrega un elemento al final.
- **Desencolar (dequeue)**: Se elimina el elemento del frente.

Ejemplo en C (Cola Simple) I

```
#include <stdio.h>
#define MAX 5

typedef struct {
    int frente, final;
    int items[MAX];
} Cola;

void encolar(Cola *q, int valor) {
    if (q->final == MAX - 1) {
        printf("La cola est  llena\n");
        return;
    }
    if (q->frente == -1) {
        q->frente = 0;
    }
```

Ejemplo en C (Cola Simple) II

```
    q->final++;
    q->items[q->final] = valor;
}

int desencolar(Cola *q) {
    if (q->frente == -1) {
        printf("La cola est  vac a\n");
        return -1;
    }
    int valor = q->items[q->frente];
    q->frente++;
    if (q->frente > q->final) {
        q->frente = q->final = -1;
    }
    return valor;
}
```

Ejercicio a Resolver

Ejercicio 2: Simulación de una fila en un banco

- Implementa un programa en C que simule una fila en un banco.
- Los clientes llegan y se forman en la fila para ser atendidos.
- Cada vez que un cliente es atendido, se desencola de la fila.
- El programa debe permitir:
  - Agregar clientes a la cola.
  - Atender a un cliente (desencolar).
  - Mostrar el cliente actual en la fila (frente).
  - Mostrar si la fila está vacía.
- Tamaño máximo de la fila: 10 clientes.

¿Qué es una Cola Circular?

- Una **cola circular** es una variante de la cola lineal en la que el último elemento apunta al primero.
- La estructura sigue la política **FIFO** (*First In, First Out*).
- Se implementa típicamente en arreglos donde el índice de inserción vuelve al inicio al llegar al final.
- Evita el desperdicio de espacio en arreglos cuando se utilizan en procesos repetitivos.

Características de una Cola Circular

- Usa dos punteros:
  - **Frente**: Apunta al primer elemento en la cola.
  - **Final**: Indica la última posición del elemento añadido.
- Cuando el final alcanza el último índice, el siguiente encolar se realiza al inicio del arreglo (si hay espacio).
- El índice se calcula como: $\text{pos} = (\text{pos} + 1) \mod \text{MAX}$.

Ejemplo de Cola Circular

- Supongamos un arreglo de tamaño 5.
- Insertamos elementos y mostramos cómo se çierra"la cola al completarse un ciclo.

Frente Final

10	20	30	40	50

¿Qué es una Cola Doble?

- Una **cola doble** (Deque) permite la inserción y eliminación de elementos por ambos extremos.
- Puede operar tanto como una pila (LIFO) o como una cola (FIFO), según el tipo de operación.
- Útil en aplicaciones donde se necesita flexibilidad de acceso en ambos extremos.

Tipos de Operaciones en una Cola Doble

- **Inserción en el frente**: Agrega un elemento al inicio de la cola.
- **Inserción en el final**: Agrega un elemento al final de la cola.
- **Eliminación desde el frente**: Remueve el primer elemento de la cola.
- **Eliminación desde el final**: Remueve el último elemento de la cola.

Ejemplo de Cola Doble (Deque)

- Supongamos una cola con operaciones en ambos extremos.
- Elementos iniciales en la cola: [10, 20, 30]
- Operaciones:
  - Insertar al frente: 5
  - Insertar al final: 40
  - Eliminar desde el frente
  - Eliminar desde el final

Frente Final

5	10	20	30

Ejemplo en C: Cola Circular I

```c
#include <stdio.h>
#define MAX 5  // Tamaño máximo de la cola

typedef struct {
    int items[MAX];
    int frente, final;
} ColaCircular;

// Inicializa la cola circular
void inicializarCola(ColaCircular *cola) {
    cola->frente = -1;
    cola->final = -1;
}

// Verifica si la cola está llena
int esLlena(ColaCircular *cola) {
```

Ejemplo en C: Cola Circular II

```
    return (cola->frente == (cola->final + 1) % MAX
}

// Verifica si la cola est  vac a
int esVacia(ColaCircular *cola) {
    return (cola->frente == -1);
}

// Encola un elemento al final de la cola
void encolar(ColaCircular *cola, int valor) {
    if (esLlena(cola)) {
        printf("La cola est  llena. No se puede en
        return;
    }
    if (esVacia(cola)) {  // Si la cola est  vac a
        cola->frente = 0;
    }
```

Ejemplo en C: Cola Circular III

```
        cola->final = (cola->final + 1) % MAX;
        cola->items[cola->final] = valor;
        printf("Encolado:-%d\n", valor);
    }

    // Desencola un elemento del frente de la cola
    int desencolar(ColaCircular *cola) {
        if (esVacia(cola)) {
            printf("La-cola-est -vac a.-No-se-puede-d
            return -1;
        }
        int valor = cola->items[cola->frente];
        if (cola->frente == cola->final) {  // La cola
            cola->frente = -1;
            cola->final = -1;
        } else {
            cola->frente = (cola->frente + 1) % MAX;
```

Ejemplo en C: Cola Circular IV

```
        }
        printf("Desencolado: %d\n", valor);
        return valor;
    }

    // Muestra el estado de la cola
    void mostrarCola(ColaCircular *cola) {
        if (esVacia(cola)) {
            printf("La cola est  vac a\n");
            return;
        }
        printf("Cola: ");
        int i = cola->frente;
        while (1) {
            printf("%d ", cola->items[i]);
            if (i == cola->final) break;
            i = (i + 1) % MAX;
```

Ejemplo en C: Cola Circular V

```
        }
        printf("\n");
    }

    int main() {
        ColaCircular cola;
        inicializarCola(&cola);

        encolar(&cola, 10);
        encolar(&cola, 20);
        encolar(&cola, 30);
        encolar(&cola, 40);
        encolar(&cola, 50);  // Cola llena despu s de
        mostrarCola(&cola);

        desencolar(&cola);
        desencolar(&cola);
```

Ejemplo en C: Cola Circular VI

```
    mostrarCola(&cola);

    encolar(&cola, 60);
    mostrarCola(&cola);

    return 0;
}
```

Ejemplo de Cola Doble (Deque) en C I

```c
#include <stdio.h>
#define MAX 5

typedef struct {
    int items[MAX];
    int front;
    int rear;
} Deque;

void initDeque(Deque* dq) {
    dq->front = -1;
    dq->rear = -1;
}

int isFull(Deque* dq) {
    return (dq->front == 0 && dq->rear == MAX - 1)
```

Ejemplo de Cola Doble (Deque) en C II

```
}

int isEmpty(Deque* dq) {
    return dq->front == -1;
}

void addFront(Deque* dq, int value) {
    if (isFull(dq)) {
        printf("Deque lleno\n");
        return;
    }
    if (dq->front == -1) {
        dq->front = dq->rear = 0;
    } else if (dq->front == 0) {
        dq->front = MAX - 1;
    } else {
        dq->front--;
```

Ejemplo de Cola Doble (Deque) en C III

```
        }
        dq->items[dq->front] = value;
    }

    void addRear(Deque* dq, int value) {
        if (isFull(dq)) {
            printf("Deque lleno\n");
            return;
        }
        if (dq->front == -1) {
            dq->front = dq->rear = 0;
        } else if (dq->rear == MAX - 1) {
            dq->rear = 0;
        } else {
            dq->rear++;
        }
        dq->items[dq->rear] = value;
```

Ejemplo de Cola Doble (Deque) en C IV

```
}

void removeFront(Deque* dq) {
    if (isEmpty(dq)) {
        printf("Deque vacio\n");
        return;
    }
    printf("Removido del frente: %d\n",
    dq->items[dq->front]);
    if (dq->front == dq->rear) {
        dq->front = dq->rear = -1;
    } else if (dq->front == MAX - 1) {
        dq->front = 0;
    } else {
        dq->front++;
    }
}
```

Ejemplo de Cola Doble (Deque) en C V

```
void removeRear(Deque* dq) {
    if (isEmpty(dq)) {
        printf("Deque vacio\n");
        return;
    }
    printf("Removido de la parte trasera: %d\n",
    dq->items[dq->rear]);
    if (dq->front == dq->rear) {
        dq->front = dq->rear = -1;
    } else if (dq->rear == 0) {
        dq->rear = MAX - 1;
    } else {
        dq->rear--;
    }
}
```

Ejemplo de Cola Doble (Deque) en C VI

```
void getFront(Deque* dq) {
    if (isEmpty(dq)) {
        printf("Deque vacio\n");
    } else {
        printf("Elemento frontal: %d\n",
        dq->items[dq->front]);
    }
}

void getRear(Deque* dq) {
    if (isEmpty(dq)) {
        printf("Deque vacio\n");
    } else {
        printf("Elemento trasero: %d\n",
        dq->items[dq->rear]);
    }
}
```

Ejemplo de Cola Doble (Deque) en C VII

```c
int main() {
    Deque dq;
    initDeque(&dq);

    addRear(&dq, 10);
    addRear(&dq, 20);
    addFront(&dq, 5);
    getFront(&dq);
    getRear(&dq);

    removeFront(&dq);
    getFront(&dq);
    removeRear(&dq);
    getRear(&dq);

    return 0;
```

Ejemplo de Cola Doble (Deque) en C VIII

```
}
```

Fin

¡Gracias por su atención!
¿Preguntas?