Algoritmi di ordinamento e strutture dati

Ecco uno schema generale basato su selection sort, insertion sort, pila, coda e struct, includendo anche il codice fornito:

Algoritmi di Ordinamento e Strutture Dati

Selection Sort

Il selection sort è un algoritmo di ordinamento che seleziona ripetutamente l'elemento minimo (o massimo) dalla parte non ordinata del vettore e lo posiziona nella parte ordinata.

Pseudocodice

```
for i = 0 to n-1
    minIndex = i
    for j = i+1 to n-1
        if v[j] < v[minIndex]
            minIndex = j
    swap v[i] and v[minIndex]
```

Implementazione in C++

```cpp
void selectionSort(int v[], int n) {
    for (int i = 0; i < n - 1; i++) {
        int minIndex = i;
        for (int j = i + 1; j < n; j++) {
            if (v[j] < v[minIndex]) {
                minIndex = j;
            }
        }
        swap(v[i], v[minIndex]);
    }
}
```

Insertion Sort

L'insertion sort è un algoritmo di ordinamento che costruisce la parte ordinata inserendo un elemento alla volta nella posizione corretta.

Pseudocodice

```
for i = 1 to n-1
    key = v[i]
    j = i - 1
    while j ≥ 0 and v[j] > key
        v[j+1] = v[j]
        j = j - 1
    v[j+1] = key
```

Implementazione in C++

```cpp
void insertionSort(int v[], int n) {
    for (int i = 1; i < n; i++) {
        int key = v[i];
        int j = i - 1;
        while (j ≥ 0 && v[j] > key) {
            v[j + 1] = v[j];
            j--;
        }
        v[j + 1] = key;
    }
}
```

Pila (Stack)

La pila è una struttura dati lineare che segue il principio LIFO (Last-In-First-Out), dove l'ultimo elemento inserito è il primo ad essere rimosso.

Operazioni principali

- `push` : Aggiunge un elemento in cima alla pila.
- `pop` : Rimuove l'elemento in cima alla pila.
- `top` : Restituisce l'elemento in cima alla pila senza rimuoverlo.
- `isEmpty` : Verifica se la pila è vuota.

Implementazione in C++ utilizzando un vettore

```
struct vettore {
    int n;      // numero massimo di elementi
    int v[10];  // vettore
    int i;      // indice
};

void push(vettore &a, int valore) {
    if (a.i < a.n) {
        a.v[a.i] = valore;
        a.i++;
    }
}

int pop(vettore &a) {
    int ris = -1;
    if (a.i > 0) {
        ris = a.v[a.i - 1];
        a.i--;
    }
    return ris;
}
```

Coda (Queue)

La coda è una struttura dati lineare che segue il principio FIFO (First-In-First-Out), dove il primo elemento inserito è il primo ad essere rimosso.

Operazioni principali

- `enqueue` : Aggiunge un elemento in fondo alla coda.
- `dequeue` : Rimuove l'elemento in testa alla coda.
- `front` : Restituisce l'elemento in testa alla coda senza rimuoverlo.
- `isEmpty` : Verifica se la coda è vuota.

Implementazione in C++ utilizzando un vettore

```
struct vettore {
    int n;      // numero massimo di elementi
    int v[10];  // vettore
```

```
    int i;      // indice
};

void push(vettore &a, int valore) {
    if (a.i < a.n) {
        a.v[a.i] = valore;
        a.i++;
    }
}

int pop(vettore &a) {
    int ris = -1;
    if (a.i > 0) {
        ris = a.v[0];
        for (int j = 0; j < a.i - 1; j++) {
            a.v[j] = a.v[j + 1];
        }
        a.i--;
    }
    return ris;
}
```

Struct

La struct in C++ è un tipo di dato composto che può contenere diverse variabili di tipi diversi.

Esempio di struct

```
struct vettore {
    int n;      // numero massimo di elementi
    int v[10];  // vettore
    int i;      // indice
};
```

Questa struct rappresenta un vettore con un numero massimo di elementi (`n`), il vettore stesso (`v`) e un indice (`i`) per tenere traccia della posizione corrente.

Funzioni Ausiliarie

Oltre agli algoritmi di ordinamento e alle strutture dati, sono state fornite diverse funzioni ausiliarie per operazioni comuni sui vettori:

- `findMax`: Trova l'elemento massimo in un vettore.
- `findIndexMax`: Trova l'indice dell'elemento massimo in un vettore.
- `findMin`: Trova l'elemento minimo in un vettore.
- `findIndexMin`: Trova l'indice dell'elemento minimo in un vettore.
- `findIndexElement`: Trova l'indice di un elemento specifico in un vettore.
- `numberElement`: Conta il numero di occorrenze di un elemento in un vettore.
- `stampa`: Stampa gli elementi di un vettore.

Queste funzioni possono essere utilizzate in combinazione con gli algoritmi di ordinamento e le strutture dati per svolgere diverse operazioni sui vettori.

Conclusioni

Gli algoritmi di ordinamento come selection sort e insertion sort sono fondamentali per ordinare gli elementi di un vettore. Le strutture dati come la pila e la coda offrono modalità specifiche per l'inserimento e la rimozione degli elementi, seguendo rispettivamente i principi LIFO e FIFO. La struct consente di creare tipi di dati composti per rappresentare strutture più complesse.

Combinando questi concetti e utilizzando le funzioni ausiliarie, è possibile risolvere una vasta gamma di problemi che coinvolgono l'elaborazione e la manipolazione dei dati.