**Ex No 1: Reading and Printing N Integer Elements of an Array**

**Aim:**

Write a Program to Read and Print an Array of N Integer Elements.

**Program:**

**#include <iostream>**

**using namespace std;**

**int main()**

**{**

**int n,a[20],i;**

**cout << "Enter the number of elements: ";**

**cin >> n;**

**for (int i = 0; i < n; i++)**

**{**

**cout << "Element " << i + 1 << ": ";**

**cin >> a[i];**

**}**

**// Display the array**

**cout << "The array elements are:\n";**

**for (int i = 0; i < n; i++)**

**{**

**cout << "Element " << i + 1 << ": " << a[i] << endl;**

**}**

**return 0;**

**}**

**Result:**

The above program was successfully compiled and executed.

**Ex No 2 Implementation of Array**

**Aim:** Write a C++ program to implement the following operations

* Input and display array elements
* Insert element at a specific position
* Delete element from a specific position
* Search for an element

**Program:**

**#include <iostream>**

**using namespace std;**

**#define MAX 100 // Maximum size of array**

**class Array**

**{**

**int a[20];**

**int size;**

**public:**

**Array()**

**{**

**size = 0;**

**}**

**void input(int n)**

**{**

**size = n;**

**cout << "Enter " << size << " elements:\n";**

**for (int i = 0; i < size; i++)**

**cin >> a[i];**

**}**

**void display()**

**{**

**cout << "Array elements: ";**

**for (int i = 0; i < size; i++)**

**cout << a[i] << " ";**

**cout << endl;**

**}**

**void insert(int pos, int element)**

**{**

**for (int i = size; i > pos; i--)**

**a[i] = a[i - 1];**

**a[pos] = element;**

**size++;**

**cout << "Element inserted.\n";**

**}**

**void remove(int pos)**

**{**

**for (int i = pos; i < size - 1; i++)**

**a[i] = a[i + 1];**

**size--;**

**cout << "Element deleted.\n";**

**}**

**void search(int element)**

**{**

**for (int i = 0; i < size; i++)**

**{**

**if (a[i] == element)**

**{**

**cout << "Element found at index " << i << endl;**

**return;**

**}**

**}**

**cout << "Element not found.\n";**

**}**

**};**

**int main()**

**{**

**Array a;**

**int n, choice, pos, val;**

**cout << "Enter number of elements: ";**

**cin >> n;**

**a.input(n);**

**do**

**{**

**cout << "\nMenu:\n";**

**cout << "1. Display\n2. Insert\n3. Delete\n4. Search\n5. Exit\n";**

**cout << "Enter choice: ";**

**cin >> choice;**

**switch (choice)**

**{**

**case 1:**

**a.display();**

**break;**

**case 2:**

**cout << "Enter position and value to insert: ";**

**cin >> pos >> val;**

**a.insert(pos, val);**

**break;**

**case 3:**

**cout << "Enter position to delete: ";**

**cin >> pos;**

**a.remove(pos);**

**break;**

**case 4:**

**cout << "Enter value to search: ";**

**cin >> val;**

**a.search(val);**

**break;**

**case 5:**

**cout << "Exiting...\n";**

**break;**

**default:**

**cout << "Invalid choice.\n";**

**}**

**} while (choice != 5);**

**return 0;**

**}**

**Result:**

The array implementation was successfully executed and verified

### Ex No : 3 Insert an Element into a Linked List

### Aim :

### Write a c++ program to inserting an element into a Linked List

### Program:

**#include <iostream>**

**using namespace std;**

**// Node structure**

**struct Node {**

**int data;**

**Node\* link;**

**};**

**// Function to create a new node**

**Node\* createNode(int value) {**

**Node\* newNode = new Node(); // dynamically create node**

**newNode->data = value;**

**newNode->link = nullptr;**

**return newNode;**

**}**

**// Function to insert at the beginning**

**void insertAtBeginning(Node\*& head, int value) {**

**Node\* newNode = createNode(value);**

**newNode->link = head;**

**head = newNode;**

**}**

**// Function to insert at the end**

**void insertAtEnd(Node\*& head, int value) {**

**Node\* newNode = createNode(value);**

**if (head == nullptr) {**

**head = newNode;**

**} else {**

**Node\* temp = head;**

**while (temp->link != nullptr) {**

**temp = temp->link;**

**}**

**temp->link = newNode;**

**}**

**}**

**// Function to insert at a specific position (1-based index)**

**void insertAtPosition(Node\*& head, int value, int position) {**

**if (position <= 0) {**

**cout << "Invalid position.\n";**

**return;**

**}**

**if (position == 1) {**

**insertAtBeginning(head, value);**

**return;**

**}**

**Node\* newNode = createNode(value);**

**Node\* temp = head;**

**for (int i = 1; temp != nullptr && i < position - 1; i++) {**

**temp = temp->link;**

**}**

**if (temp == nullptr) {**

**cout << "Position out of range.\n";**

**return;**

**}**

**newNode->link = temp->link;**

**temp->link = newNode;**

**}**

**// Function to display the list**

**void displayList(Node\* head) {**

**cout << "Linked List: ";**

**while (head != nullptr) {**

**cout << head->data << " -> ";**

**head = head->link;**

**}**

**cout << "NULL\n";**

**}**

**// Main function**

**int main() {**

**Node\* head = nullptr;**

**int choice, value, position;**

**do {**

**cout << "\n1. Insert at Beginning\n2. Insert at End\n3. Insert at Position\n4. Display List\n5. Exit\n";**

**cout << "Enter your choice: ";**

**cin >> choice;**

**switch (choice) {**

**case 1:**

**cout << "Enter value to insert at beginning: ";**

**cin >> value;**

**insertAtBeginning(head, value);**

**break;**

**case 2:**

**cout << "Enter value to insert at end: ";**

**cin >> value;**

**insertAtEnd(head, value);**

**break;**

**case 3:**

**cout << "Enter value to insert: ";**

**cin >> value;**

**cout << "Enter position: ";**

**cin >> position;**

**insertAtPosition(head, value, position);**

**break;**

**case 4:**

**displayList(head);**

**break;**

**case 5:**

**cout << "Exiting...\n";**

**break;**

**default:**

**cout << "Invalid choice.\n";**

**}**

**} while (choice != 5);**

**return 0;**

**}**

Result: The above program was executed and verified successfully.

**Ex.No:4 Delete an Element from Singly Linked List**

**Aim:** Write a C++ program to delete an element from the Linked List

* Delete the **first node** (head)
* Delete the **last node**
* Delete a **node with a specific value**

**Program:**

#include <iostream>

using namespace std;

// Node structure for singly linked list

struct Node {

int data;

Node\* next;

};

// Function to create a new node

Node\* createNode(int data) {

Node\* newNode = new Node();

newNode->data = data;

newNode->next = nullptr;

return newNode;

}

// Function to insert a node at the end

void insertAtEnd(Node\*& head, int data) {

Node\* newNode = createNode(data);

if (head == nullptr) {

head = newNode;

return;

}

Node\* temp = head;

while (temp->next != nullptr)

temp = temp->next;

temp->next = newNode;

}

// Function to delete the first node

void deleteAtBeginning(Node\*& head) {

if (head == nullptr) {

cout << "List is empty.\n";

return;

}

Node\* temp = head;

head = head->next;

delete temp;

}

// Function to delete the last node

void deleteAtEnd(Node\*& head) {

if (head == nullptr) {

cout << "List is empty.\n";

return;

}

if (head->next == nullptr) {

delete head;

head = nullptr;

return;

}

Node\* temp = head;

while (temp->next->next != nullptr)

temp = temp->next;

delete temp->next;

temp->next = nullptr;

}

// Function to delete a node by value

void deleteByValue(Node\*& head, int value) {

if (head == nullptr) {

cout << "List is empty.\n";

return;

}

if (head->data == value) {

deleteAtBeginning(head);

return;

}

Node\* temp = head;

while (temp->next != nullptr && temp->next->data != value)

temp = temp->next;

if (temp->next == nullptr) {

cout << "Value " << value << " not found.\n";

return;

}

Node\* nodeToDelete = temp->next;

temp->next = temp->next->next;

delete nodeToDelete;

}

// Function to display the linked list

void displayList(Node\* head) {

cout << "Linked List: ";

while (head != nullptr) {

cout << head->data << " -> ";

head = head->next;

}

cout << "NULL\n";

}

int main() {

Node\* head = nullptr;

// Inserting elements

insertAtEnd(head, 10);

insertAtEnd(head, 20);

insertAtEnd(head, 30);

insertAtEnd(head, 40);

cout << "Original List:\n";

displayList(head);

deleteAtBeginning(head);

cout << "After deleting from beginning:\n";

displayList(head);

deleteAtEnd(head);

cout << "After deleting from end:\n";

displayList(head);

deleteByValue(head, 20);

cout << "After deleting value 20:\n";

displayList(head);

return 0;

}

**Ex No 5 :** **Inserting Element in a Doubly Linked List**

**Aim:**

Write a c++ program to insert a new element into a doubly linked list

* At the beginning
* At the end
* After a given node (by key)

**Program:**

#include <iostream>

using namespace std;

// Node structure for Doubly Linked List

struct Node {

int data;

Node\* prev;

Node\* next;

};

// Function to create a new node

Node\* createNode(int data) {

Node\* newNode = new Node();

newNode->data = data;

newNode->prev = newNode->next = nullptr;

return newNode;

}

// Function to insert at the beginning

void insertAtBeginning(Node\*& head, int data) {

Node\* newNode = createNode(data);

if (head != nullptr) {

newNode->next = head;

head->prev = newNode;

}

head = newNode;

}

// Function to insert at the end

void insertAtEnd(Node\*& head, int data) {

Node\* newNode = createNode(data);

if (head == nullptr) {

head = newNode;

return;

}

Node\* temp = head;

while (temp->next != nullptr)

temp = temp->next;

temp->next = newNode;

newNode->prev = temp;

}

// Function to insert after a given node

void insertAfter(Node\* head, int key, int data) {

Node\* temp = head;

while (temp != nullptr && temp->data != key)

temp = temp->next;

if (temp == nullptr) {

cout << "Node with data " << key << " not found.\n";

return;

}

Node\* newNode = createNode(data);

newNode->next = temp->next;

newNode->prev = temp;

if (temp->next != nullptr)

temp->next->prev = newNode;

temp->next = newNode;

}

// Function to display the list

void displayList(Node\* head) {

Node\* temp = head;

cout << "Doubly Linked List: ";

while (temp != nullptr) {

cout << temp->data << " <-> ";

temp = temp->next;

}

cout << "NULL\n";

}

int main() {

Node\* head = nullptr;

insertAtEnd(head, 10);

insertAtEnd(head, 20);

insertAtEnd(head, 30);

cout << "After inserting at end:\n";

displayList(head);

insertAtBeginning(head, 5);

cout << "After inserting at beginning:\n";

displayList(head);

insertAfter(head, 20, 25);

cout << "After inserting 25 after 20:\n";

displayList(head);

return 0;

}

Result:

### Ex No 6 Operations of Stack Using Array

### Aim: Write a C++program to implement stack operations using array.

#include <iostream>

using namespace std;

#define MAX 100

int stack[MAX];

int top = -1;

// Push operation

void push(int value) {

if (top >= MAX - 1) {

cout << "Stack Overflow! Cannot push " << value << "\n";

return;

}

stack[++top] = value;

cout << value << " pushed to stack.\n";

}

// Pop operation

void pop() {

if (top < 0) {

cout << "Stack Underflow! Nothing to pop.\n";

return;

}

cout << stack[top--] << " popped from stack.\n";

}

// Display operation

void display() {

if (top < 0) {

cout << "Stack is empty.\n";

return;

}

cout << "Stack elements (top to bottom): ";

for (int i = top; i >= 0; i--)

cout << stack[i] << " ";

cout << "\n";

}

int main() {

int choice, value;

do {

cout << "\n--- Stack Menu ---\n";

cout << "1. Push\n2. Pop\n3. Display\n4. Exit\n";

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

cout << "Enter value to push: ";

cin >> value;

push(value);

break;

case 2:

pop();

break;

case 3:

display();

break;

case 4:

cout << "Exiting...\n";

break;

default:

cout << "Invalid choice!\n";

}

} while (choice != 4);

return 0;

}