



COMSATS University Islamabad, Vehari Campus

Department of Computer Science

Class: BCS-SP22(A)

Submission Deadline: 9 Oct 2023

Subject: Data Structures and Algorithms-Lab

Instructor: Yasmeen Jana

Reg. No: SP22-BCS-035

Name : MUFEEZ ASLAM

Activity 01

```
#include <iostream>
```

```
struct Node {
```

```
    int data;
```

```
    Node* next;
```

```
};
```

```
void displayLinkedListInfo(Node* head) {
```

```
    Node* ptr = head;
```

```
    std::cout << "***head address:** " << &head << std::endl;
```

```
    std::cout << "head content:" << head << std::endl;
```

```

while (ptr != nullptr) {

    std::cout << "*** ptr address :***" << ptr << std::endl;

    std::cout << " ptr content :" << ptr << std::endl;

    std::cout << "ptr data :" << ptr->data << std::endl;

    std::cout << "Ptr :" << ptr << std::endl;

    if (ptr->next != nullptr) {

        std::cout << "Ptr - >next:" << ptr->next << std::endl;

        std::cout << "Ptr - >data:" << ptr->next->data << std::endl;

    }

    ptr = ptr->next;

}
}

```

```

int main() {

    Node* head = new Node{1, nullptr};

    head->next = new Node{2, nullptr};

    head->next->next = new Node{20, nullptr};

    head->next->next->next = new Node{30, nullptr};

    displayLinkedListInfo(head);

    while (head != nullptr) {

        Node* temp = head;

        head = head->next;

        delete temp;

    }
}

```

```

return 0;

}

```

The screenshot shows the Programiz Online C++ Compiler interface. The code editor on the left contains a C++ program for a linked list. The output window on the right displays the execution results, showing the linked list structure and the data values.

```

main.cpp
1 #include <iostream>
2
3 struct Node {
4     int data;
5     Node* next;
6 };
7
8 void displayLinkedListInfo(Node* head) {
9     Node* ptr = head;
10
11     std::cout << "**head address:** " << &head << std::endl;
12     std::cout << "head content:" << head << std::endl;
13
14     while (ptr != nullptr) {
15         std::cout << "**ptr address:**" << ptr << std::endl;
16         std::cout << "ptr content:" << ptr << std::endl;
17         std::cout << "ptr data:" << ptr->data << std::endl;
18         std::cout << "Ptr:" << ptr << std::endl;
19         if (ptr->next != nullptr) {
20             std::cout << "Ptr ->next:" << ptr->next << std::endl;
21             std::cout << "Ptr ->data:" << ptr->next->data << std::endl;
22         }
23     }

```

Output:

```

/tmp/fGMNirWpshk.o
The linked list is
1 2 20 30
**head address:** 0x7ffe8ca20a88
head content:0x12c2eb0
**ptr address:**0x12c2eb0
ptr content :0x12c2eb0
ptr data :1
Ptr :0x12c2eb0
Ptr ->next:0x12c2ed0

ptr data :2
Ptr :0x12c2ed0
Ptr ->next:0x12c2ef0
Ptr ->data:20

ptr data :20
Ptr :0x12c2ef0
Ptr ->next:0x12c2f10

ptr data :30
Ptr :0x12c2f10

```

Activity 02

```

#include <iostream>

using namespace std;

struct Node {

    int data;

    Node* next;

};

```

```

Node* createNode(int data) {

    Node* newNode = new Node;

    newNode->data = data;

    newNode->next = nullptr;

    return newNode;

}

```

```

class LinkedList {

public:

    Node* head;

    LinkedList() : head(nullptr) {}

```

```

void insertAtBeginning(int data) {

    Node* newNode = createNode(data);

    newNode->next = head;

    head = newNode;

    cout << "Insertion at the beginning successful." << endl;

}

```

```

void insertAtEnd(int data) {

    Node* newNode = createNode(data);

    if (!head) {

        head = newNode;

        cout << "Insertion at the end successful." << endl;

        return;

    }

    Node* current = head;

    while (current->next) {

        current = current->next;

```

```

    }

    current->next = newNode;

    cout << "Insertion at the end successful." << endl;
}

void deleteNode(int data) {

    if (!head) {

        cout << "List is empty. Deletion not possible." << endl;

        return;

    }

    if (head->data == data) {

        Node* temp = head;

        head = head->next;

        delete temp;

        cout << "Deletion successful." << endl;

        return;

    }

    Node* current = head;

    while (current->next) {

        if (current->next->data == data) {

            Node* temp = current->next;

            current->next = current->next->next;

            delete temp;

            cout << "Deletion successful." << endl;

            return;

        }

        current = current->next;

    }

    cout << "Element not found. Deletion not possible." << endl;

}

```

```

void reverse() {

    Node* prev = nullptr;

    Node* current = head;

    Node* nextNode = nullptr;

    while (current) {

        nextNode = current->next;

        current->next = prev;

        prev = current;

        current = nextNode;

    }

    head = prev;

    cout << "Reversal successful." << endl;

}

```

```

void display() {

    Node* current = head;

    while (current) {

        cout << current->data << " -> ";

        current = current->next;

    }

    cout << "nullptr" << endl;

}

};

```

```

int main() {

    LinkedList singleLinkedList;

    while (true) {

        cout << "Which linked list you want:" << endl;

```

```
cout << "1: Single" << endl;
```

```
cout << "2: Double" << endl;
```

```
cout << "3: Circular" << endl;
```

```
int listChoice;
```

```
cin >> listChoice;
```

```
if (listChoice == 1) {
```

```
    int choice;
```

```
    while (true) {
```

```
        cout << "\nSingle Linked List Operations:" << endl;
```

```
        cout << "1: Insertion" << endl;
```

```
        cout << "2: Deletion" << endl;
```

```
        cout << "3: Display" << endl;
```

```
        cout << "4: Reverse" << endl;
```

```
        cout << "5: Seek" << endl;
```

```
        cout << "6: Exit" << endl;
```

```
        cin >> choice;
```

```
        if (choice == 1) {
```

```
            int insertionChoice;
```

```
            cout << "Insertion Options:" << endl;
```

```
            cout << "1: Insertion at Beginning" << endl;
```

```
            cout << "2: Insertion at End" << endl;
```

```
            cin >> insertionChoice;
```

```
            if (insertionChoice == 1) {
```

```
                int data;
```

```
                cout << "Enter data: ";
```

```
                cin >> data;
```

```

        singleLinkedList.insertAtBeginning(data);

    } else if (insertionChoice == 2) {

        int data;

        cout << "Enter data: ";

        cin >> data;

        singleLinkedList.insertAtEnd(data);

    }

    } else if (choice == 2) {

        int data;

        cout << "Enter data to delete: ";

        cin >> data;

        singleLinkedList.deleteNode(data);

    } else if (choice == 3) {

        singleLinkedList.display();

    } else if (choice == 4) {

        singleLinkedList.reverse();

    } else if (choice == 5) {

        // Handle seek option

    } else if (choice == 6) {

        // Exit the program

        return 0;

    } else {

        cout << "Invalid choice. Please enter a valid option." << endl;

    }

}

}

}

}

return 0;

}

```


The screenshot shows the Programiz C++ Online Compiler web application. At the top, there's a navigation bar with the Programiz logo and "C++ Online Compiler". Below this is a header section featuring advertisements for Intuit Quickbooks and a "C++ Certification" button. The main workspace is divided into two panes. The left pane, titled "main.cpp", contains C++ code for a linked list implementation, including headers, namespace declarations, struct definitions, and functions like createNode, insertAtBeginning, and LinkedList constructor. The right pane, titled "Output", displays the execution results, showing the path to the compiled executable, a menu of operations (Single, Double, Circular), and the output of the "Insertion at Beginning" operation.

```

#include <iostream>
using namespace std;

struct Node {
    int data;
    Node* next;
};

Node* createNode(int data) {
    Node* newNode = new Node;
    newNode->data = data;
    newNode->next = nullptr;
    return newNode;
}

class LinkedList {
public:
    Node* head;

    LinkedList() : head(nullptr) {}

    void insertAtBeginning(int data) {
        Node* newNode = createNode(data);

```

Output:

```

/tmp/fGMNrvpshk.o
Which linked list you want:
1: Single
2: Double
3: Circular
1
Single Linked List Operations:
1: Insertion
2: Deletion
3: Display
4: Reverse
5: Seek
6: Exit
1
Insertion Options:
1: Insertion at Beginning
2: Insertion at End
1
Enter data: 1
Insertion at the beginning successful.

Single Linked List Operations:
1: Insertion

```