|  |  |
| --- | --- |
| download | *Assignment no 2*  Name:M.Haseeb khan  Reg no:FA20-BCS-042  Section:B  COMSATS University Islamabad,Vehari Campus |

Answer no1:

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

};

void displayLinkedList(Node\* head) {

cout << "The linked list is: ";

Node\* ptr = head;

while (ptr != NULL) {

cout << ptr->data << " ";

ptr = ptr->next;

}

cout << endl << "\*\*head address: " << head << endl;

cout << "--------------------------" << endl;

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

cout << "--------------------------" << endl;

cout << "\*\*ptr address:\*\* " << &head << endl;

cout << "--------------------------" << endl;

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

cout << "----------------------" << endl;

ptr = head;

while (ptr != NULL) {

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

cout << "----------------------" << endl;

cout << "ptr: " << ptr << endl;

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

ptr = ptr->next;

}

}

int main() {

Node\* head = new Node();

Node\* second = new Node();

Node\* third = new Node();

Node\* fourth = new Node();

head->data = 1;

head->next = second;

second->data = 2;

second->next = third;

third->data = 20;

third->next = fourth;

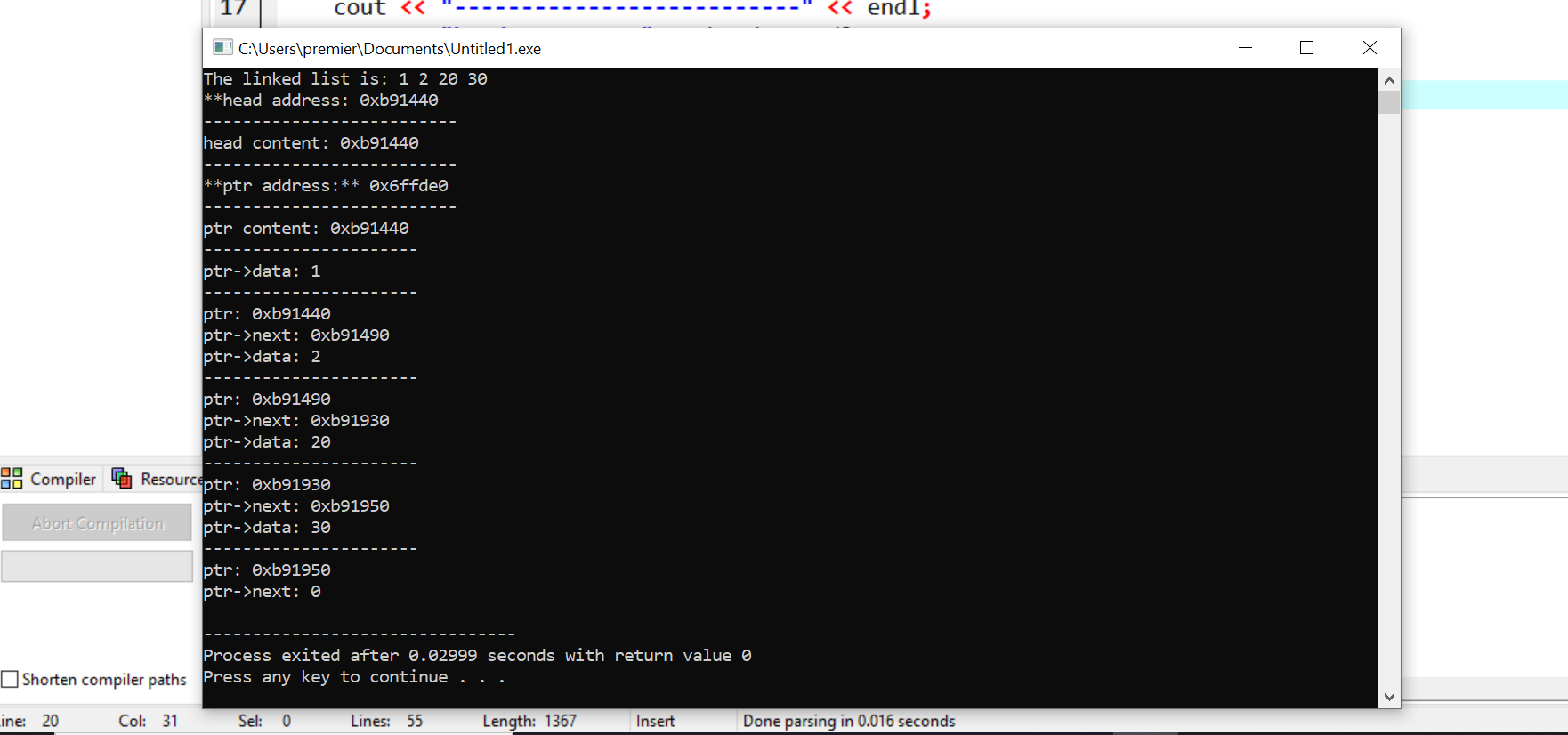
fourth->data = 30;

fourth->next = NULL;

displayLinkedList(head);

return 0;

}

Output:

Answer no 2:

#include <iostream>

#include <conio.h>

using namespace std;

struct Node {

int data;

Node\* next;

Node\* prev;

Node(int val) : data(val), next(NULL), prev(NULL) {}

};

class LinkedList {

private:

Node\* head;

Node\* tail;

bool isCircular;

public:

LinkedList(bool circular = false) : head(NULL), tail(NULL), isCircular(circular) {}

void insertAtBeginning(int value) {

Node\* newNode = new Node(value);

if (isCircular) {

if (head == NULL) {

newNode->next = newNode;

} else {

newNode->next = head;

Node\* lastNode = head;

while (lastNode->next != head) {

lastNode = lastNode->next;

}

lastNode->next = newNode;

}

head = newNode;

} else {

newNode->next = head;

head = newNode;

}

cout << "Inserted successfully at the beginning." << endl;

}

void insertAtEnd(int value) {

Node\* newNode = new Node(value);

if (isCircular) {

if (head == NULL) {

newNode->next = newNode;

head = newNode;

} else {

newNode->next = head;

Node\* lastNode = head;

while (lastNode->next != head) {

lastNode = lastNode->next;

}

lastNode->next = newNode;

}

} else {

if (head == NULL) {

head = newNode;

tail = newNode;

} else {

tail->next = newNode;

tail = newNode;

}

}

cout << "Inserted successfully at the end." << endl;

}

void insertAfterValue(int value, int target) {

Node\* newNode = new Node(value);

Node\* current = head;

while (current != NULL) {

if (current->data == target) {

newNode->next = current->next;

current->next = newNode;

cout << "Inserted successfully after " << target << "." << endl;

return;

}

current = current->next;

}

cout << "Value " << target << " not found in the list." << endl;

}

void display() {

Node\* current = head;

cout << "The items present in the list are: ";

if (current == NULL) {

cout << "Empty";

} else {

if (isCircular) {

do {

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

current = current->next;

} while (current != head);

} else {

while (current != NULL) {

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

current = current->next;

}

}

}

cout << endl;

}

void reverse() {

Node\* prev = NULL;

Node\* current = head;

Node\* next = NULL;

while (current != NULL) {

next = current->next;

current->next = prev;

prev = current;

current = next;

}

head = prev;

cout << "List reversed." << endl;

}

void seekValue(int value) {

Node\* current = head;

int position = 0;

while (current != NULL) {

if (current->data == value) {

cout << "Value " << value << " found at position " << position << "." << endl;

return;

}

current = current->next;

position++;

}

cout << "Value " << value << " not found in the list." << endl;

}

void deleteList() {

Node\* current = head;

while (current != NULL) {

Node\* next = current->next;

delete current;

current = next;

}

head = NULL;

cout << "List deleted." << endl;

}

~LinkedList() {

deleteList();

}

};

int main() {

int choice;

bool isCircular = false;

LinkedList list(isCircular);

do {

cout << "Operations on List.." << 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;

cout << "Enter your choice: ";

cin >> choice;

switch (choice) {

case 1:

int insertChoice;

cout << "1. Insertion at the beginning" << endl;

cout << "2. Insertion at the end" << endl;

cout << "3. Insertion at a specific data node" << endl;

cout << "Enter your choice: ";

cin >> insertChoice;

int insertValue;

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

cin >> insertValue;

switch (insertChoice) {

case 1:

list.insertAtBeginning(insertValue);

break;

case 2:

list.insertAtEnd(insertValue);

break;

case 3:

int insertTarget;

cout << "Enter the target value: ";

cin >> insertTarget;

list.insertAfterValue(insertValue, insertTarget);

break;

default:

cout << "Invalid choice!" << endl;

break;

}

break;

case 2:

break;

case 3:

list.display();

break;

case 4:

list.reverse();

break;

case 5:

int seekValue;

cout << "Enter the value to seek: ";

cin >> seekValue;

list.seekValue(seekValue);

break;

case 6:

cout << "Exiting the program..." << endl;

list.deleteList();

exit(0);

default:

cout << "Invalid choice!" << endl;

break;

}

cout << "Press any key to continue...";

cin.ignore();

cin.get();

} while (choice != 6);

return 0;

}

Output:

