**Bài 20:**

1. Xác định các lớp, các thuộc tính, các phương thức của lớp và mô tả chức năng của từng lớp, từng phương thức.

Các lớp bao gồm:

- Lớp Node

- Lớp Double LinkedList

- Lớp App

1. **Lớp Node**

* Các thuộc tính: value, địa chỉ node kế tiếp, địa chỉ node phía trước
* Các phương thức khỏi tạo: Khởi tạo giá trị ban đầu cho một node bao gồm (datatype value, Node\* next, Node\* prev)
* Các phương thức getter, setter: Lấy giá trị từ các biến private và chỉnh sửa giá trị các biến.
* Phương thức gán: Gán giá trị một node bằng một node khác.

1. **Lớp DoubleLinkedList**

* Các thuộc tính: Địa chỉ node đầu tiên, địa chỉ node cuối cùng
* Các phương thức:

- Phương thức khởi tạo: Khởi tạo một List các Node.

- Phương thức **append**(T value): Thêm 1 node vào cuối danh sách

- Phương thức **displayDoubleLinkedList**() const: hiển thị danh sách từ đầu đến cuối DoubleLinkedList

- Phương thức **displayRerevesDoubleLinkedList**() const: Hiển thị danh sách từ cuối về đầu của DoubleLinkedList

- Phương thức **removeDuplicate**(): Xóa hết các phần từ xuất hiện từ 2 lần trở lên ra khỏi list để mỗi phần từ chỉ xuất hiện 1 lần.

- Phương thức **transformList**() : Tác động 1 hàm fun(x) vào từng phần tử x của list để biến đổi list.

- Phương thức **sortList():** Sắp xếp doubleLinkedList theo thứ tự tăng dần.

- Phương thức **insertSorted():** Chèn thêm 1 phần tử vào danh sách đã sắp xếp tăng dần để được danh sách tăng dần

1. **Cài đặt các lớp và hàm main bằng C++**

|  |
| --- |
| **Cài đặt lớp Node:** |
| #include <iostream>  #include <set>  #include <vector>  #include <algorithm>  using namespace std;  template <class T>  class Node {  T value;  Node \*next, \*prev;  public:  Node() : next(NULL), prev(NULL) {}  Node(T value, Node \*next = NULL, Node \*prev = NULL)  : value(value), next(next), prev(prev) {}  void setValue(T value) {  this->value = value;  }  void setNext(Node<T> \*N = NULL) {  next = N;  }  void setPrev(Node<T> \*N = NULL) {  prev = N;  }  Node<T>\* getNext() const {  return next;  }  Node<T>\* getPrev() const {  return prev;  }  T getValue() const {  return value;  }  Node<T>& operator=(const Node& x) {  if (this != &x) {  this->value = x.value;  this->next = x.next;  this->prev = x.prev;  }  return \*this;  }  }; |

|  |
| --- |
| **Cài đặt lớp DoubleLinkedList** |
| template <class T>  class DoubleLinkedList {  private:  Node<T>\* head;  Node<T>\* tail;  public:  DoubleLinkedList() : head(NULL), tail(NULL) {}  void append(T value) {  Node<T>\* newNode = new Node<T>(value);  if (head == NULL) {  head = tail = newNode;  } else {  tail->setNext(newNode);  newNode->setPrev(tail);  tail = newNode;  }  }  void displayDoubleLinkedList() const {  Node<T>\* current = head;  while (current != NULL) {  cout << current->getValue();  if (current->getNext() != NULL) cout << " <=> ";  current = current->getNext();  }  cout << endl;  }  void displayReverseDoubleLinkedList() const {  Node<T>\* current = tail;  while (current != NULL) {  cout << current->getValue();  if (current->getPrev() != NULL) cout << " <=> ";  current = current->getPrev();  }  cout << endl;  }  void removeDuplicate() {  set<T> seen;  Node<T>\* current = head;  while (current != NULL) {  if (seen.find(current->getValue()) != seen.end()) {  Node<T>\* toDelete = current;  if (current->getPrev() != NULL) {  current->getPrev()->setNext(current->getNext());  } else {  head = current->getNext();  }  if (current->getNext() != NULL) {  current->getNext()->setPrev(current->getPrev());  } else {  tail = current->getPrev();  }  current = current->getNext();  delete toDelete;  } else {  seen.insert(current->getValue());  current = current->getNext();  }  }  }  void transformList() {  Node<T>\* current = head;  while (current != NULL) {  T newValue = (current->getValue() \* current->getValue() - 7 \* current->getValue() + 3) % 100;  current->setValue(newValue);  current = current->getNext();  }  removeDuplicate();  }  void sortList() {  vector<T> values;  Node<T>\* current = head;  while (current != NULL) {  values.push\_back(current->getValue());  current = current->getNext();  }  sort(values.begin(), values.end());  current = head;  for (size\_t i = 0; i < values.size(); i++) {  current->setValue(values[i]);  current = current->getNext();  }  }  void insertSorted(T value) {  Node<T>\* newNode = new Node<T>(value);  if (head == NULL || head->getValue() >= value) {  newNode->setNext(head);  if (head != NULL) {  head->setPrev(newNode);  }  head = newNode;  if (tail == NULL) {  tail = head;  }  return;  }  Node<T>\* current = head;  while (current->getNext() != NULL && current->getNext()->getValue() < value) {  current = current->getNext();  }  newNode->setNext(current->getNext());  if (current->getNext() != NULL) {  current->getNext()->setPrev(newNode);  }  current->setNext(newNode);  newNode->setPrev(current);  if (newNode->getNext() == NULL) {  tail = newNode;  }  }  ~DoubleLinkedList() {  Node<T>\* current = head;  while (current != NULL) {  Node<T>\* next = current->getNext();  delete current;  current = next;  }  }  }; |

|  |
| --- |
| **Cài đặt lớp App và hàm main()** |
| class App {  private:  DoubleLinkedList<int> list;  public:  void run() {  int choice;  do {  cout << "\n--------------MENU----------------" << endl;  cout << "1. Nhap 1 day so nguyen vao list" << endl;  cout << "2. In danh sach double list" << endl;  cout << "3. Xoa di nhung phan tu trung nhau" << endl;  cout << "4. Bien doi tung phan tu x thành (x ^ 2 - 7x + 3) % 100 sau dó loai bo cac so trung lap." << endl;  cout << "5. Sap xep tang dan" << endl;  cout << "6. Nhap x và chen vào list de danh sách van tang dan." << endl;  cout << "7. Xuat day theo chieu nguoc lai" << endl;  cout << "8. Thoat" << endl;  cout << "Lua chon cua ban: ";  cin >> choice;    switch (choice) {  case 1: {  int n;  cout << "Nhap so phan tu: ";  cin >> n;  cout << "Nhap cac phan tu: ";  for (int i = 0; i < n; i++) {  int val;  cin >> val;  list.append(val);  }  break;  }  case 2:  list.displayDoubleLinkedList();  break;  case 3:  list.removeDuplicate();  break;  case 4:  list.transformList();  break;  case 5:  list.sortList();  break;  case 6: {  int x;  cout << "Nhap gia tri x can chen: ";  cin >> x;  list.insertSorted(x);  break;  }  case 7:  list.displayReverseDoubleLinkedList();  break;  case 8:  cout << "Thoat chuong trinh." << endl;  break;  default:  cout << "Nhap lua chon không hop le. Vui long chon lai!" << endl;  }  } while (choice != 8);  }  };  int main() {  App app;  app.run();  return 0;  } |