|  |
| --- |
| #pragma once  #include<string>  #include<iostream>  using namespace std;  class node  {  string name;  int finger;  node \*next;  node \*prev;  public:  node()  {  name = "";  finger = 0;  next = nullptr;  prev = nullptr;  }  ~node()  {  }  int getfinger()  {  return finger;  }  void setfinger(int finger)  {  this->finger = finger;  }  string getname()  {  return name;  }  void setname(string name)  {  this->name = name;  }  node \*getnext()  {  return next;  }  void setnext(node \*next)  {  this->next = next;  }  void setprev(node \*prev)  {  this->prev = prev;  }  node \*getprev()  {  return prev;  }  }; |

|  |
| --- |
| #pragma once  #include"node.h"  #include<iostream>  #include<string>  using namespace std;  class list  {  int size;  node \*headnode;  node \*currentnode;  public:  list()  {  size = 0;  headnode = nullptr;  currentnode = nullptr;  }  ~list()  {  }  void insertion();  void deletion();  void update();  void start();  void next();  void back();  void tail();  int find();  void get();  void length();  void display();  void exit();  }; |

|  |
| --- |
| #include "list.h"  #include"node.h"  #include<string>  #include<iostream>  using namespace std;  int main()  {  list l;  int f;  int c;  do  {  cout << endl  << "$$$ MENU $$$" << endl  << "1: insertion" << endl  << "2: deletion" << endl  << "3: update" << endl  << "4: start" << endl  << "5: next" << endl  << "6: back" << endl  << "7: tail" << endl  << "8: find" << endl  << "9: get" << endl  << "10: length" << endl  << "11: display list" << endl  << "12: exit" << endl  << "enter your choice: ";  cin >> c;  switch (c)  {  case 1:  {  l.insertion();  break;  }  case 2:  {  l.deletion();  break;  }  case 3:  {  l.update();  break;  }  case 4:  {  l.start();  break;  }  case 5:  {  l.next();  break;  }  case 6:  {  l.back();  break;  }  case 7:  {  l.tail();  break;  }  case 8:  {  f = l.find();  if (f == 1)  cout << "value has successfully been found" << endl;  else if (f == 0)  cout << "value has not been found" << endl;  break;  }  case 9:  {  l.get();  break;  }  case 10:  {  l.length();  break;  }  case 11:  {  l.display();  break;  }  case 12:  {  l.exit();  break;  }  default:cout << "enter correct choice" << endl;  }  } while (c != 12);  system("pause");  return 0;  }  void list::insertion()  {  string n;  int x, a, b, c;  node \*newnode = new node();  cout << "enter your name: ";  getline(cin, n, '.');  cout << "enter no of finger(must be less than 5): ";  cin >> x;  if ((x <= 5) && (x > 0))  {  newnode->setnext(nullptr);  newnode->setprev(nullptr);  newnode->setname(n);  newnode->setfinger(x);  size++;  if (headnode == nullptr)  {  headnode = newnode;  currentnode = newnode;  }  else  {  cout << endl;  cout << "$$$ MENU $$$" << endl  << "1: by location" << endl  << "2: by value" << endl  << "0: exit" << endl  << "enter your choice";  cin >> a;  switch (a)  {  case 1:  {  cout << endl;  cout << "### MENU ###" << endl  << "a: after the current location" << endl  << "b: before the current location" << endl  << "z: exit" << endl  << "enter your choice";  cin >> b;  switch (b)  {  case 'a':  {  if (currentnode->getnext() != nullptr)  {  newnode->setprev(currentnode);  newnode->setnext(currentnode->getnext());  currentnode->setnext(newnode);  (newnode->getnext())->setprev(newnode);  }  else if (currentnode->getnext() == nullptr)  {  newnode->setprev(currentnode);  newnode->setnext(currentnode->getnext());  currentnode->setnext(newnode);  //(newnode->getnext())->setprev(newnode);  }  else  {  cout << "error" << endl;  }  break;  }  case 'b':  {  newnode->setnext(currentnode);  newnode->setprev(currentnode->getprev());  currentnode->setprev(newnode);  (newnode->getprev())->setnext(newnode);  break;  }  case 'z':  {  break;  }  default:cout << "enter correct choice" << endl;  }  break;  }  case 2:  {  cout << endl;  cout << "### MENU ###" << endl  << "a: after the value" << endl  << "b: before the value" << endl  << "z: exit" << endl  << "enter your choice";  cin >> c;  switch (c)  {  case 'a':  {  if (find() == 1)  {  newnode->setprev(currentnode);  newnode->setnext(currentnode->getnext());  currentnode->setnext(newnode);  (newnode->getnext())->setprev(newnode);  }  else  {  cout << "value not found" << endl;  }  break;  }  case 'b':  {  if (find() == 1)  {  newnode->setnext(currentnode);  newnode->setprev(currentnode->getprev());  currentnode->setprev(newnode);  (newnode->getprev())->setnext(newnode);  }  else  {  cout << "value not found" << endl;  }  break;  }  case 'z':  {  break;  }  default:cout << "enter correct choice" << endl;  }  break;  }  case 0:  {  break;  }  default:cout << "enter correct choice" << endl;  }  }  }  else  {  cout << "the no of fingers must be less then 5" << endl;  }  }  void list::deletion()  {  if (headnode != nullptr)  {  int x;  cout << "$$$ MENU $$$" << endl  << "1: by value" << endl  << "2: headnode" << endl  << "3: last node" << endl  << "0: exit" << endl  << "enter your choice";  cin >> x;  switch (x)  {  case 1:  {  if (find() == 1)  {  if (headnode != currentnode)  {  node \*temp = currentnode;  (currentnode->getprev())->setnext(currentnode->getnext());  (currentnode->getnext())->setprev(currentnode->getprev());  currentnode = currentnode->getnext();  delete temp;  size--;  }  else if (currentnode == headnode)  {  node \*temp = headnode;  headnode = headnode->getnext();  delete temp;  headnode->setprev(nullptr);  size--;  }  }  else  {  cout << "value not found" << endl;  }  break;  }  case 2:  {  node \*temp = headnode;  headnode = headnode->getnext();  delete temp;  headnode->setprev(nullptr);  size--;  break;  }  case 3:  {  node \*temp = currentnode;  (currentnode->getnext())->setprev(currentnode->getprev());  (currentnode->getprev())->setnext(currentnode->getnext());  currentnode = currentnode->getprev();  size--;  break;  }  case 0:break;  default:cout << "enter correct choice" << endl;  }  }  else  {  cout << "error: list has not been created" << endl;  }  }  void list::update()  {  if (headnode != nullptr)  {  string n;  int x;  if (find() == 1)  {  cout << "enter your name";  getline(cin, n, '.');  cout << "enter no of fingers: ";  cin >> x;  if (x <= 5)  {  currentnode->setfinger(x);  currentnode->setname(n);  cout << "value has been updated" << endl;  }  else  {  cout << "no of fingers must be less than 5" << endl;  }  }  else  {  cout << "your are already on the start" << endl;  }  }  else  {  cout << "error: list has not been created" << endl;  }  }  void list::start()  {  if (headnode != nullptr)  {  if (currentnode != headnode)  {  currentnode = headnode;  cout << "now you are on start node" << endl;  cout << "now the name is: " << currentnode->getname() << endl;  cout << "now the no of fingers are: " << currentnode->getfinger() << endl;  }  else  {  cout << "your are already on the start" << endl;  }  }  else  {  cout << "error: list has not been created" << endl;  }  }  void list::next()  {  if (headnode != nullptr)  {  if (currentnode->getnext() != nullptr)  {  currentnode = currentnode->getnext();  cout << "you are currently on next node" << endl;  cout << "now the no of fingers are: " << currentnode->getfinger() << endl;  cout << "now the name is: " << currentnode->getname() << endl;  }  else  {  cout << "you are already on the lastnode" << endl;  }  }  else  {  cout << "error: list has not been created" << endl;  }  }  void list::back()  {  if (headnode != nullptr)  {  if ((currentnode->getprev() != nullptr) || (currentnode != headnode))  {  currentnode = currentnode->getprev();  cout << "now you are currently on back node" << endl;  cout << "now the no of fingers are : " << currentnode->getfinger() << endl;  cout << "now the name is: " << currentnode->getname() << endl;  }  else  {  cout << "you are already on the start node" << endl;  }  }  else  {  cout << "error: list has not been created" << endl;  }  }  void list::tail()  {  if (headnode != nullptr)  {  if (currentnode->getnext() != nullptr)  {  while (currentnode->getnext() != nullptr)  {  currentnode = currentnode->getnext();  }  cout << "now you are currently on end node" << endl;  cout << "now the no of fingers are: " << currentnode->getfinger() << endl;  cout << "now the name is: " << currentnode->getname() << endl;  }  else  {  cout << "you are already on the tail or endnode" << endl;  }  }  else  {  cout << "error: list has not been created" << endl;  }  }  int list::find()  {  int flag = 3;  string n;  if (headnode != nullptr)  {  cout << "enter a name: ";  getline(cin, n, '.');  node \*temp = headnode;  while (temp != nullptr)  {  if (n == temp->getname())  {  flag = 1;  currentnode = temp;  break;  }  else  {  flag = 0;  temp = temp->getnext();  }  }  }  else  {  cout << "error: list has not been created" << endl;  }  return flag;  }  void list::get()  {  if (headnode != nullptr)  {  cout << "the curent index of the node is: " << currentnode << endl  << "the no of fingers on current index are: " << currentnode->getfinger() << endl;  cout << "the name on current index is: " << currentnode->getname() << endl;  }  else  {  cout << "error: list has not been created" << endl;  }  }  void list::length()  {  if (headnode != nullptr)  {  cout << "the length or the size of list is: " << size << endl;  }  else  {  cout << "error: list has not been created" << endl;  }  }  void list::display()  {  if (headnode != nullptr)  {  node \*temp;  temp = headnode;  while (temp != nullptr)  {  cout << "no of finger are: " << temp->getfinger() << endl;  cout << "the name is: " << temp->getname() << endl;  temp = temp->getnext();  }  }  else  {  cout << "error: list has not been created" << endl;  }  }  void list::exit()  {  cout << "program ended" << endl;  } |