**Лабораторная работа №2**

Выполнил: Лосев Данил ЕПИ-4-23

# Задание:

1. Реализовать рассмотренный в Лекции №2 готовый код функции построения линейного списка void Postroenie () в удобном для Вас редакторе кода на языке программирования С++.
2. На основе алгоритмов из лекции 2, созданных с помощью схем Дональда Кнута, написать соответствующие коды функций *с комментариями* и реализовать их в редакторе кода:
3. Самостоятельно написать коды функций:

* Вывод линейного списка на экран
* Поиск элемента линейного списка по его значению
* Подсчет и вывод на экран количества звеньев в построенном Вами линейном списке
* Построение кольцевого списка
* Вывод на консоль кольцевого списка
* Все функции объединить в единый листинг с помощью меню.

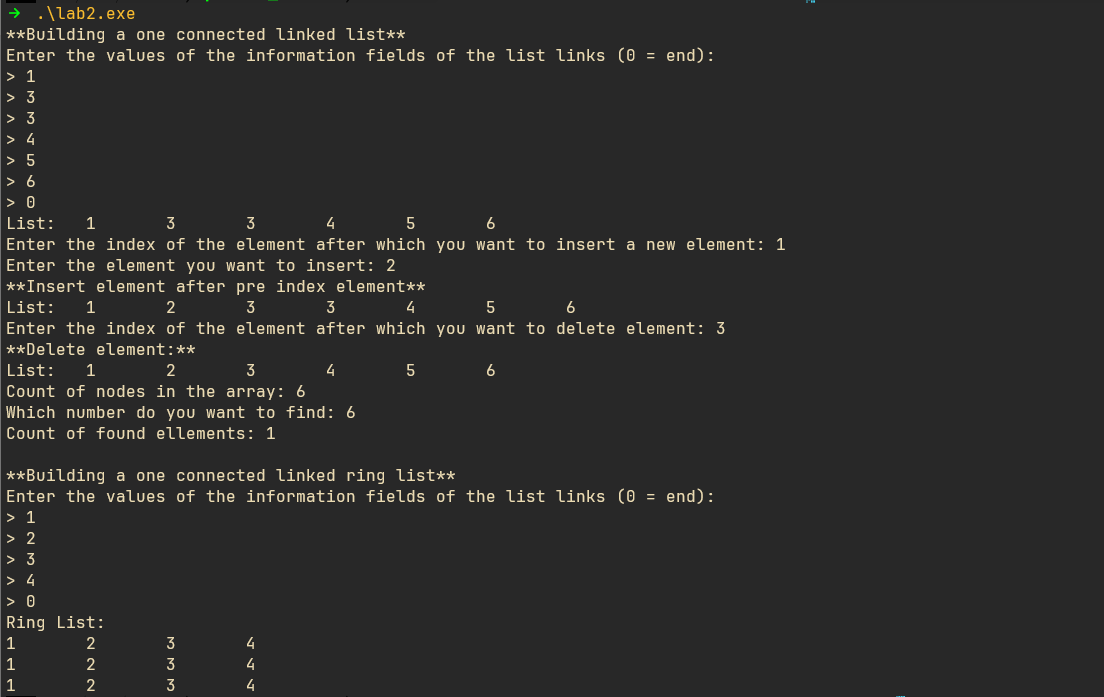
1. Оформить отчет в электронном виде.

# Листинг:

|  |
| --- |
| #include <iostream>  // Структура узла, которая хранит значение и указатель на следующий элемент  struct Node  {  int value; // Значение узла  Node \*next; // Указатель на следующий узел  };  // Прототипы функций для работы с односвязным списком  void posroenie(Node \*&fPHead); // Функция построения списка  void vstavka(Node \*&fPHead, int preIndex, int fElement); // Вставка нового элемента в список  void udalenie(Node \*&fPHead, int preIndex); // Удаление элемента из списка  void printList(Node \*fPHead); // Печать списка  int findNode(Node \*fPHead, int fEllement); // Поиск элемента в списке  int countOfNodes(Node \*fPHead); // Подсчёт количества узлов в списке  void makeARingList(Node \*&fPHead); // Создание кольцевого списка  void printRingList(Node \*fPHead, int countOfRing = 1); // Печать кольцевого списка  int main()  {  // Объявляю указатели на начало списка и кольцевого списка  Node \*HeadOfOneConnectedList; // Указатель на односвязный список  Node \*HeadOfOneConnectedRingList; // Указатель на кольцевой список  int index = 0; // Индекс для вставки или удаления элемента  int element = 0; // Элемент, который буду вставлять  int findEllement = 0; // Элемент для поиска  int countOfFoundEllements = 0; // Количество найденных элементов  // Построение односвязного списка  posroenie(HeadOfOneConnectedList);  printList(HeadOfOneConnectedList); // Печать списка  // Вставка нового элемента после определённого индекса  std::cout << "Enter the index of the element after which you want to insert a new element: ";  std::cin >> index;  std::cout << "Enter the element you want to insert: ";  std::cin >> element;  vstavka(HeadOfOneConnectedList, index, element); // Вставка элемента  printList(HeadOfOneConnectedList); // Печать списка  // Удаление элемента после определённого индекса  std::cout << "Enter the index of the element after which you want to delete element: ";  std::cin >> index;  udalenie(HeadOfOneConnectedList, index); // Удаление элемента  printList(HeadOfOneConnectedList); // Печать списка после удаления  int countOfElements = countOfNodes(HeadOfOneConnectedList); // Подсчёт узлов  // Вывод количества узлов в списке  std::cout << "Count of nodes in the array: " << countOfElements << std::endl;  // Поиск элемента в списке  std::cout << "Which number do you want to find: ";  std::cin >> findEllement;  countOfFoundEllements = findNode(HeadOfOneConnectedList, findEllement); // Поиск элемента  std::cout << "Count of found ellements: " << countOfFoundEllements << std::endl;  std::cout << std::endl;  // Построение кольцевого списка  makeARingList(HeadOfOneConnectedRingList);  printRingList(HeadOfOneConnectedRingList, 3); // Печать кольцевого списка  return 0;  }  // Функция построения односвязного списка  void posroenie(Node \*&fPHead)  {  std::cout << "\*\*Building a one connected linked list\*\*" << std::endl;  fPHead = new (Node); // Создаю первый узел  Node \*curNode = fPHead; // Текущий узел для итерации  (\*curNode).next = NULL; // Инициализация указателя на следующий узел  int ellement; // Переменная для ввода значения узла  std::cout << "Enter the values of the information fields of the list links (0 = end):" << '\n' << "> ";  std::cin >> ellement;  // Цикл для ввода элементов списка до тех пор, пока не введу 0  while (ellement != 0)  {  (\*curNode).next = new (Node); // Создаю новый узел  curNode = (\*curNode).next; // Перехожу на следующий узел  (\*curNode).value = ellement; // Записываю значение узла  (\*curNode).next = NULL; // Указатель на следующий узел равен NULL  std::cout << "> ";  std::cin >> ellement; // Ввожу следующий элемент  }  }  // Функция вставки нового узла после заданного индекса  void vstavka(Node \*&fPHead, int preIndex, int fElement)  {  std::cout << "\*\*Insert element after pre index element\*\*" << std::endl;  Node \*curNode = fPHead; // Начинаю с первого узла  // Прохожу по списку до узла с нужным индексом  for (int i = 0; i < preIndex; i++)  {  curNode = (\*curNode).next;  }  Node \*newNode = new (Node); // Создаю новый узел  (\*newNode).value = fElement; // Заполняю его значение  (\*newNode).next = (\*curNode).next; // Устанавливаю связь со следующим узлом  (\*curNode).next = newNode; // Привязываю новый узел к текущему  }  // Функция удаления узла из списка  void udalenie(Node \*&fPHead, int preIndex)  {  std::cout << "\*\*Delete element:\*\*" << std::endl;  Node \*curNode = fPHead; // Начинаю с первого узла  // Прохожу по списку до узла перед нужным для удаления  for (int i = 0; i < preIndex; i++)  {  curNode = (\*curNode).next;  }  // Проверяю, есть ли узел для удаления  if ((\*curNode).next != NULL)  {  Node \*temp = (\*curNode).next; // Временная переменная для хранения узла  (\*curNode).next = (\*(\*curNode).next).next; // Пропускаю удаляемый узел  delete temp; // Удаляю узел  }  else  {  std::cout << "You can't delete a non-existent node" << std::endl; // Ошибка, если узла нет  }  }  // Функция печати списка  void printList(Node \*fPHead)  {  Node \*curNode = fPHead; // Начинаю с первого узла  std::cout << "List: ";  while ((\*curNode).next != NULL) // Прохожу по узлам, пока не дойдём до конца  {  curNode = (\*curNode).next;  std::cout << '\t' << (\*curNode).value; // Вывожу значение узла  }  std::cout << std::endl;  }  // Функция поиска количества элементов в списке  int findNode(Node \*fPHead, int fEllement)  {  Node \*curNode = fPHead; // Начинаю с первого узла  int counfOfEl = 0; // Счётчик найденных элементов  while ((\*curNode).next != NULL)  {  curNode = (\*curNode).next;  if ((\*curNode).value == fEllement) // Сравниваю значение узла с искомым  {  counfOfEl++; // Увеличиваю счётчик, если элемент найден  }  }  return counfOfEl; // Возвращаю количество найденных элементов  }  // Функция подсчёта узлов в списке  int countOfNodes(Node \*fPHead)  {  Node \*curNode = fPHead; // Начинаю с первого узла  int count = 0; // Счётчик узлов  while ((\*curNode).next != NULL)  {  curNode = (\*curNode).next;  count++; // Увеличиваем счётчик для каждого узла  }  return count; // Возвращаю количество узлов  }  // Функция создания кольцевого списка  void makeARingList(Node \*&fPHead)  {  std::cout << "\*\*Building a one connected linked ring list\*\*" << std::endl;  fPHead = new (Node); // Создаю первый узел  Node \*curNode = fPHead; // Текущий узел  (\*curNode).next = NULL;  int ellement;  std::cout << "Enter the values of the information fields of the list links (0 = end):" << '\n' << "> ";  std::cin >> ellement;  while (ellement != 0)  {  (\*curNode).next = new (Node); // Создаю новый узел  curNode = (\*curNode).next;  (\*curNode).value = ellement;  (\*curNode).next = (\*fPHead).next; // Ссылка на первый узел кольцевого списка  std::cout << "> ";  std::cin >> ellement;  }  }  // Функция печати кольцевого списка  void printRingList(Node \*fPHead, int countOfRing)  {  Node \*curNode = fPHead; // Начинаем с первого узла  std::cout << "Ring List:" << std::endl;  for (int i = 0; i < countOfRing; i++)  {  do  {  curNode = (\*curNode).next;  std::cout << (\*curNode).value << '\t'; // Выводим значения узлов  } while ((\*curNode).next != (\*fPHead).next); // Цикл по узлам кольца  std::cout << std::endl;  }  } |

# Блок схемы:

# Скриншоты результатов работы программы:



# Ответы на вопросы:

1. **К какой категории структур данных относятся односвязные списки?**

Односвязные списки относятся к **динамическим структурам данных**.

1. **Достоинства и недостаток односвязных списков?**

Достоинства односвязных списков:

1. Динамическое управление памятью
2. Быстрая вставка и удаление элементов
3. Отсутствие ограничения на размер

Недостаток односвязных списков:

1. Медленный доступ к элементам
2. **С помощью чего в курсе описаны алгоритмы работы со структурами данных?**

С помощью схем Дональда Кнута

1. **Напишите операцию присваивания адреса в ячейку указателя t из ячейки указателя phead?**

t = phead;

1. **Заполните поля созданного звена списка с помощью команд С++:**

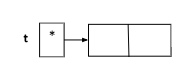
struct node

{

int value;

node \*next;

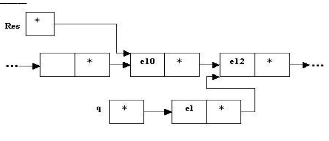
};



(\*t).value = el;

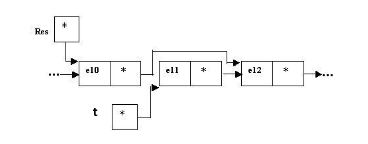
(\*t). next = NULL;

1. **Напишите оператор, с помощью которого можно в поле адреса звена с указателем q поместить адрес звена, следующим за звеном с указателем Res**



(\*t). next = (\*Res). next;

1. **Напишите оператор, с помощью которого можно перенаправить указатель Res на последующее звено e12**



(\*Res). next = (\*(\*Res). next). next);

1. **Куда можно вставить звено в списке?**

Звено в односвязном списке можно вставить в начало, конец, после или перед любым узлом списка.

1. **Напишите оператор исключения удаляемого элемента из списка**

(\*Res). next = (\*(\*Res). next). next);