

МИНОБРНАУКИ РОССИИ

*Федеральное государственное бюджетное образовательное учреждение высшего образования*

***«МИРЭА – Российский технологический университет»***

**РТУ МИРЭА**

|  |  |
| --- | --- |
| Отчет по выполнению практического задания №6 | |
| **Тема:** Однонаправленный динамический список | |
| Дисциплина Структуры и алгоритмы обработки данных | |
| Выполнил студент | Истратенков М. И. |
| группа | ИКБО-01-20 |

**Москва 2021**

**Содержание**

1. [Постановка задачи 3](#_bookmark0)
2. [Определение операций над списком 3](#_bookmark1)
   1. [Определение структуры узла однонаправленного списка 3](#_bookmark2)
   2. [Изображение процесса выполнения операций на списке 4](#_bookmark3)
   3. [Используемая в операциях структура данных 14](#_bookmark4)
3. [Код программы 14](#_bookmark5)

[ВЫВОДЫ 16](#_bookmark6)

[СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ 16](#_bookmark7)

## Постановка задачи

Требуется реализовать программу решения следующих задач варианта

№11 по использованию линейного однонаправленного списка:

* 1. Информационная часть узла содержит целые однозначные и двузначные числа.
  2. Разработать функцию для создания исходного списка, используя функцию вставки нового узла перед первым узлом.
  3. Разработать функцию вывода списка.
  4. Разработать функцию, которая создает массив А из 10 указателей на элемент списка и включает в список элемента массива с индексом i числа списка L, которые начинаются с цифры, равной i. Включение в конец списка. Однозначные числа включаются в список массива с индексом 0.
  5. Разработать функцию, которая удаляет список L.
  6. Разработать функцию, которая создает список L, включая в него списки массива А последовательно от списка с индексом 0 до списка с индексом 9.
  7. В основной программе выполнить тестирование каждой функции.

## Определение операций над списком

### Определение структуры узла однонаправленного списка

Согласно варианту №11 в качестве информационной части узла списка используются целые однозначные и двузначные числа.

### Изображение процесса выполнения операций на списке

* + 1. Вставка нового узла перед первым:

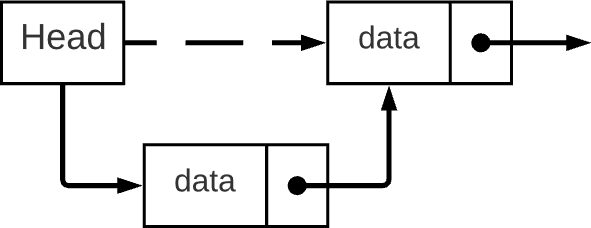


Рис. 1 Изображение вставки узла перед первым

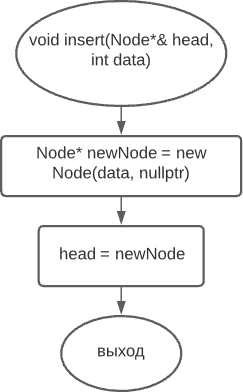


Рис. 2 Алгоритм вставки узла перед первым

Таблица 1 Набор тестов для функции

|  |  |  |  |
| --- | --- | --- | --- |
| Номер теста | Входные данные | Ожидаемый результат | Результат выполнения программы |
| 1 | Длина списка: 1  Значения: 79  Вставка: 2 | 2 79 |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 2 | Длина списка: 2  Значения: 43 47  Вставка: 0 | 0 43 47 |  |
| 3 | Длина списка: 3  Значения: 45 93 6  Вставка: 1 | 1 45 93 6 |  |

* + 1. Вставка нового узла в конец списка:

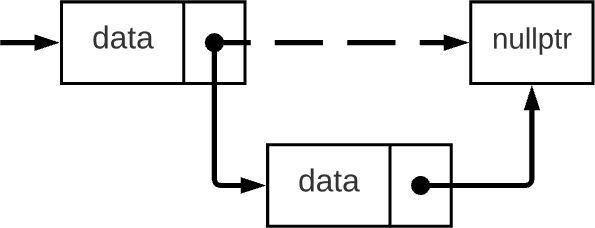


Рис. 3 Изображение вставки узла в конец списка

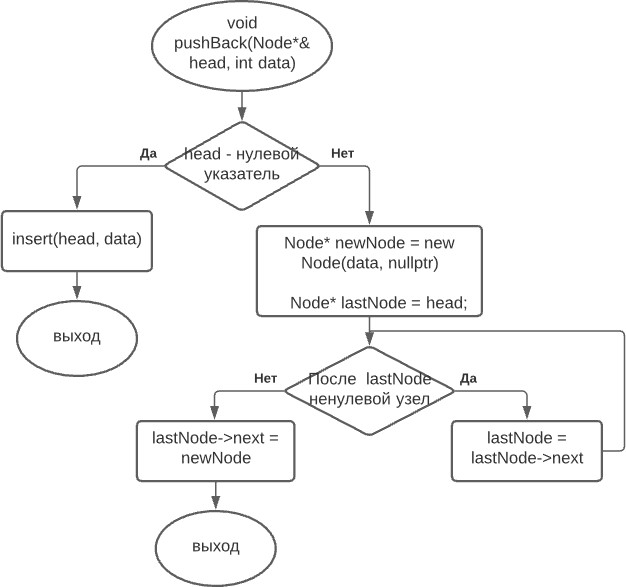


Рис. 4 Алгоритм вставки узла в конец списка

Таблица 2 Набор тестов для функции

|  |  |  |  |
| --- | --- | --- | --- |
| Номер теста | Входные данные | Ожидаемый результат | Результат выполнения программы |
| 1 | Длина списка: 1  Значения: 10  Вставка: 1 | 10 1 |  |
| 2 | Длина списка: 2  Значения: 6 5  Вставка: 2 | 6 5 2 |  |
| 3 | Длина списка: 3  Значения: 6 0 93  Вставка: 3 | 6 0 93 3 |  |

* + 1. Создание списка (посредством вставки перед первым узлом):

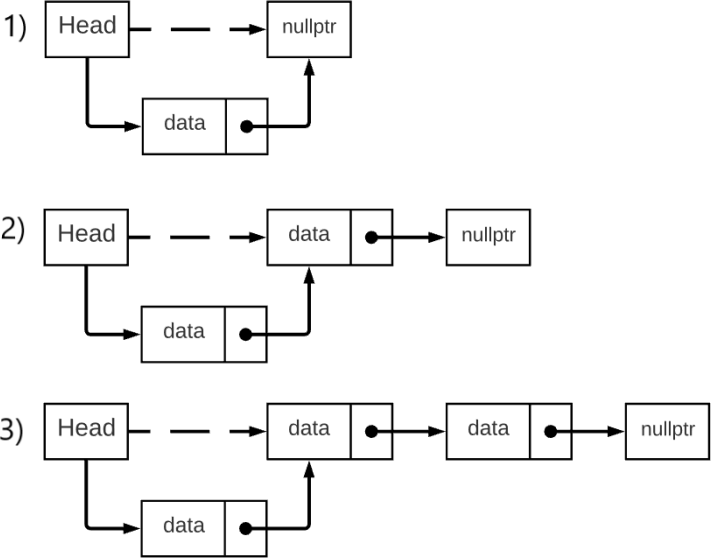


Рис. 5 Изображение создания списка на основе вставки перед первым узлом

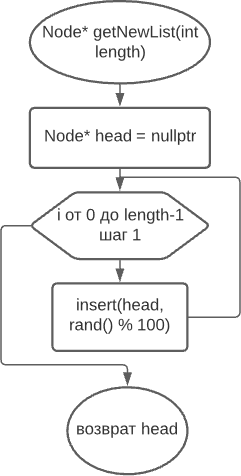


Рис. 6 Алгоритм создания списка на основе вставки перед первым узлом

Таблица 3 Набор тестов для функции

|  |  |  |  |
| --- | --- | --- | --- |
| Номер теста | Входные данные | Ожидаемый результат | Результат выполнения программы |
| 1 | Длина списка: 2  Значения: 1 2 | 2 1 |  |
| 2 | Длина списка: 3  Значения: 1 2 3 | 3 2 1 |  |
| 3 | Длина списка: 4  Значения: 1 2 3 4 | 4 3 2 1 |  |

* + 1. Включение в список элемента массива с индексом i числа списка L, которые начинаются с цифры, равной i:

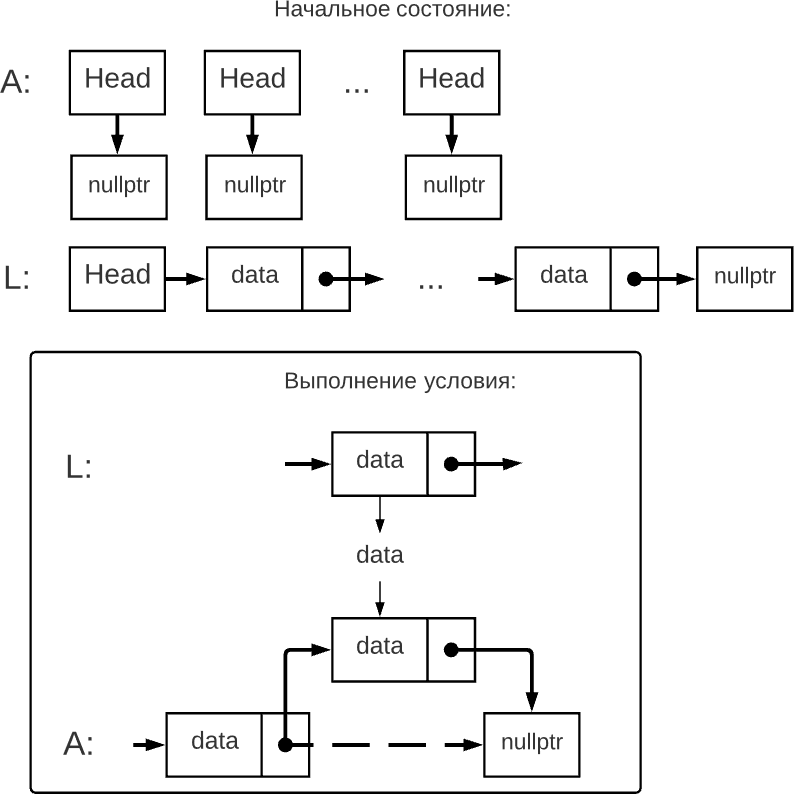


Рис. 7 Изображение включения числа из списка в список массива

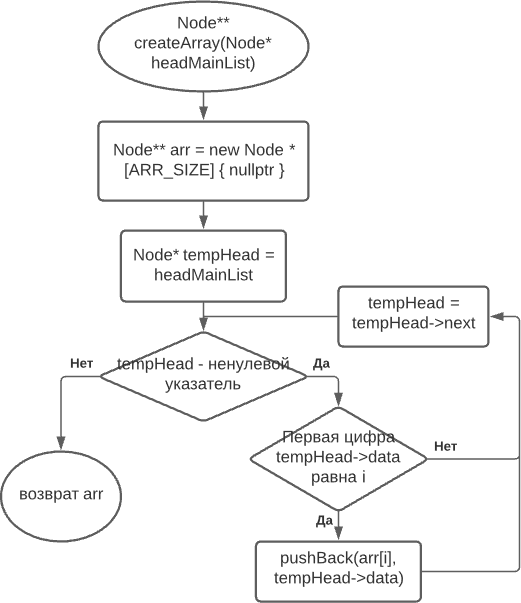


Рис. 8 Алгоритм включения числа из списка в список массива

Таблица 4 Набор тестов для функции

|  |  |  |  |
| --- | --- | --- | --- |
| Номер теста | Входные данные | Ожидаемый результат | Результат выполнения программы |
| 1 | Длина | A[0]: - |  |
|  | списка: 3 | A[1]: - |
|  | Значения: 65  30 20 | A[2]: 20  A[3]: 30 |
|  |  | A[4]: - |
|  |  | A[5]: - |
|  |  | A[6]: 65 |
|  |  | A[7]: - |
|  |  | A[8]: - |
|  |  | A[9]: - |
| 2 | Длина | A[0]: 7 |  |
|  | списка: 5 | A[1]: - |
|  | Значения: 7  38 97 64 92 | A[2]: -  A[3]: 38 |
|  |  | A[4]: - |
|  |  | A[5]: - |
|  |  | A[6]: 64 |
|  |  | A[7]: - |
|  |  | A[8]: - |
|  |  | A[9]: 97 92 |
| 3 | Длина | A[0]: - |  |
|  | списка: 5 | A[1]: - |
|  | Значения: 86  90 80 52 80 | A[2]: -  A[3]: - |
|  |  | A[4]: - |
|  |  | A[5]: 52 |
|  |  | A[6]: - |
|  |  | A[7]: - |
|  |  | A[8]: 86 80 80 |
|  |  | A[9]: 90 |

* + 1. Создание списка из списков массива A последовательно:

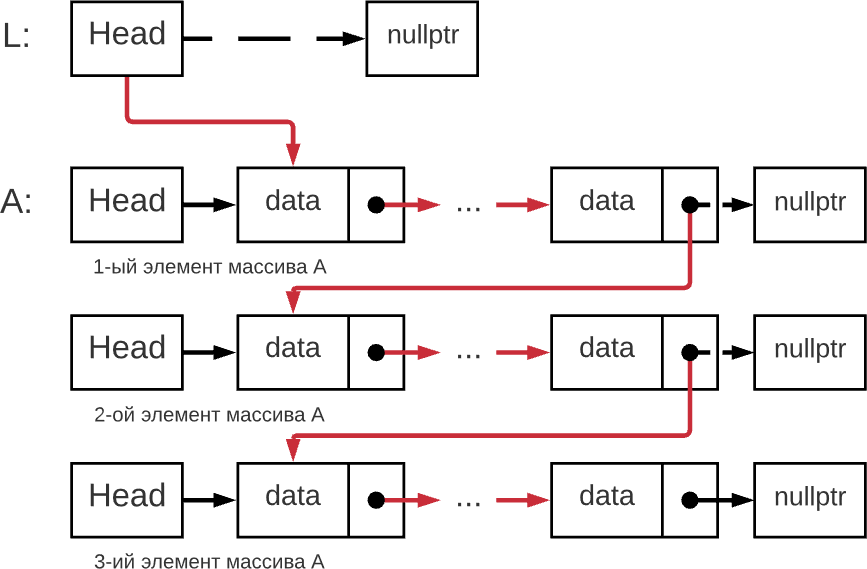


Рис. 9 Изображение создания списка из списков массива

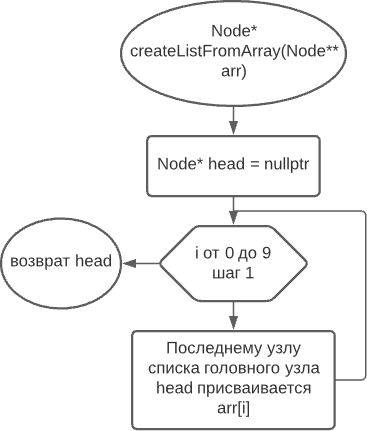


Рис. 10 Алгоритм создания списка из списков массива

Таблица 5 Набор тестов для функции

|  |  |  |  |
| --- | --- | --- | --- |
| Номер теста | Входные данные | Ожидаемый результат | Результат выполнения программы |
| 1 | Длина  списка: 3 Значения: 60 49 40 | 49 40 60 |  |
| 2 | Длина  списка: 5 Значения: 0 71 38 6  98 | 0 6 38 71 98 |  |
| 3 | Длина  списка: 5 Значения: 53 68 12 77  26 | 12 26 53 68  77 |  |

На рис. 9 красным цветом помечены связи в создаваемом линейном списке L.

### Используемая в операциях структура данных

Структура данных – однонаправленный линейный список – состоит из узлов, каждый из которых включает информационную часть и указатель на следующий узел:

struct Node {

int data; Node\* next;

Node(int data, Node\* next)

: data(data), next(next) {};

};

## Код программы

#include <iostream> #include <ctime>

const int ARR\_SIZE = 10; struct Node {

int data; Node\* next;

Node(int data, Node\* next)

: data(data), next(next) {};

};

void insert(Node\*&, int); void pushBack(Node\*&, int); Node\* getNewList(int); void printList(Node\*);

void deleteList(Node\*&);

Node\*\* createArray(Node\*);

Node\* createListFromArray(Node\*\*); void pushBackNode(Node\*&, Node\*);

int main()

{

srand(static\_cast<unsigned int>(time(0)));

int length;

std::cout << "List size = "; std::cin >> length;

Node\* head = getNewList(length); std::cout << "\nInitial "; printList(head);

Node\*\* arr = createArray(head); std::cout << "\nContents of array A\n"; for (int i = 0; i < ARR\_SIZE; ++i) {

std::cout << i << " ";

printList(arr[i]);

}

std::cout << "\n";

Node\* newHead = createListFromArray(arr); std::cout << "Resulting "; printList(newHead);

deleteList(head); deleteList(newHead); delete[] arr;

return 0;

}

void insert(Node\*& head, int data) { Node\* newNode = new Node(data, head); head = newNode;

}

void pushBack(Node\*& head, int data) { if (!head) {

insert(head, data); return;

}

Node\* newNode = new Node(data, nullptr); Node\* lastNode = head;

while (lastNode->next) lastNode = lastNode->next;

lastNode->next = newNode;

}

Node\* getNewList(int length) { Node\* head = nullptr;

for (int i = 0; i < length; ++i) insert(head, rand() % 100);

return head;

}

void printList(Node\* head) { std::cout << "List: "; if (!head) {

std::cout << "empty\n"; return;

}

std::cout << head->data;

for (Node\* temp = head->next; temp; temp = temp->next) std::cout << " -> " << temp->data;

std::cout << "\n";

}

Node\*\* createArray(Node\* headMainList) {

Node\*\* arr = new Node \* [ARR\_SIZE] { nullptr }; for (int i = 0; i < ARR\_SIZE; ++i) {

for (

Node\* tempHead = headMainList; tempHead;

tempHead = tempHead->next

) {

if (tempHead->data / 10 == i) pushBack(arr[i], tempHead->data);

}

}

return arr;

}

void deleteList(Node\*& head) { Node\* next;

for (Node\* current = head; current; current = next) { next = current->next;

delete current;

}

head = nullptr;

}

void pushBackNode(Node\*& head, Node\* node) { if (!head) {

head = node; return;

}

Node\* lastNode = head; while (lastNode->next)

lastNode = lastNode->next; lastNode->next = node;

}

Node\* createListFromArray(Node\*\* arr) { Node\* head = nullptr;

for (int i = 0; i < ARR\_SIZE; ++i) pushBackNode(head, arr[i]);

return head;

}

# ВЫВОДЫ

В ходе практической работы был разработан однонаправленный динамический список, получены знания и практические навыки управления однонаправленным динамическим списком; реализованы необходимые функции взаимодействия со списком, включая задания индивидуального варианта. Каждая выполняющаяся над списком операция прошла тестирование.

# СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

1. Кораблин Ю.П., Сыромятников В.П., Скворцова Л.А. Учебно-методическое пособие Структуры и алгоритмы обработки данных, М.:МИРЭА, 2020
2. Никлаус Вирт Алгоритмы и структуры данных. Классика программирования – М.:ДМК Пресс, 2016. — 272 с.
3. Круз Р. Л. Структуры данных и проектирование программ / пер. с англ. — 3-е издание / Р.Л. Круз. – М.:Лаборатория знаний, 2017. — 768 с.
4. Альфред В. Ахо, Джон Э. Хопкрофт, Джеффри Д. Ульман. Структуры данных и алгоритмы М.:Вильямс, 2016 — 400 с.