Министерство образования и науки Российской Федерации Федеральное агентство по образованию Федеральное государственное бюджетное образовательное учреждение высшего образования «Вятский государственный университет»

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Лабораторная работа №2

по курсу «Технология программирования»

# «Исследование состояния динамической памяти при работе с элементарными структурами данных»

Выполнил студент группы ИВТ-21\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Птахова А.М/

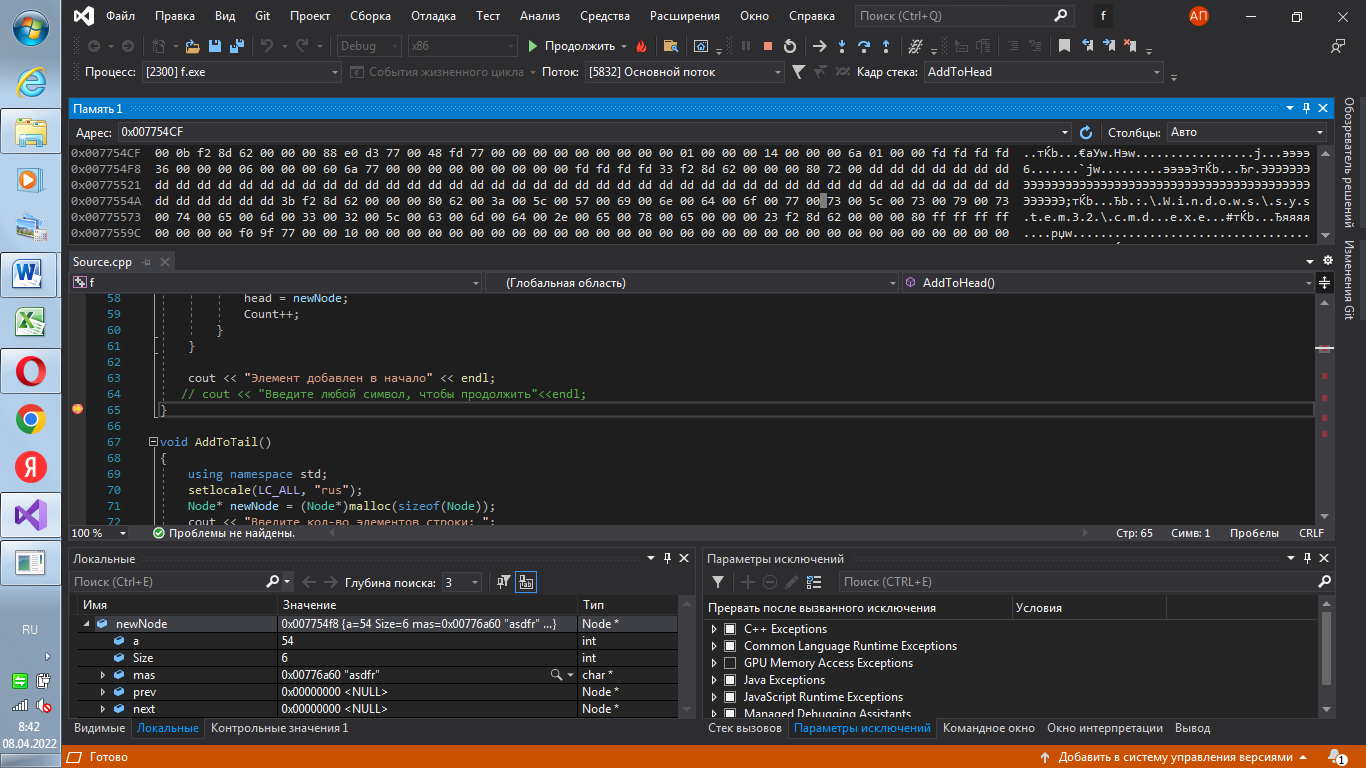
Проверила \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/ Долженкова М.Л/

Киров 2022

**Задание**: Разработать консольное приложение  для манипулирования с двусвязным списком, содержащем строку и целое число. Исследовать дамп памяти при вставке и удалении элементов структуры с учетом функцию malloc для выделения и free для освобождения  участков памяти. Доказать отсутствие утечки памяти  во время работы приложения.

**Работа программы:**

**1 Занесение элемента**



- размер структуры

- индикатор начала участка памяти

- целое число

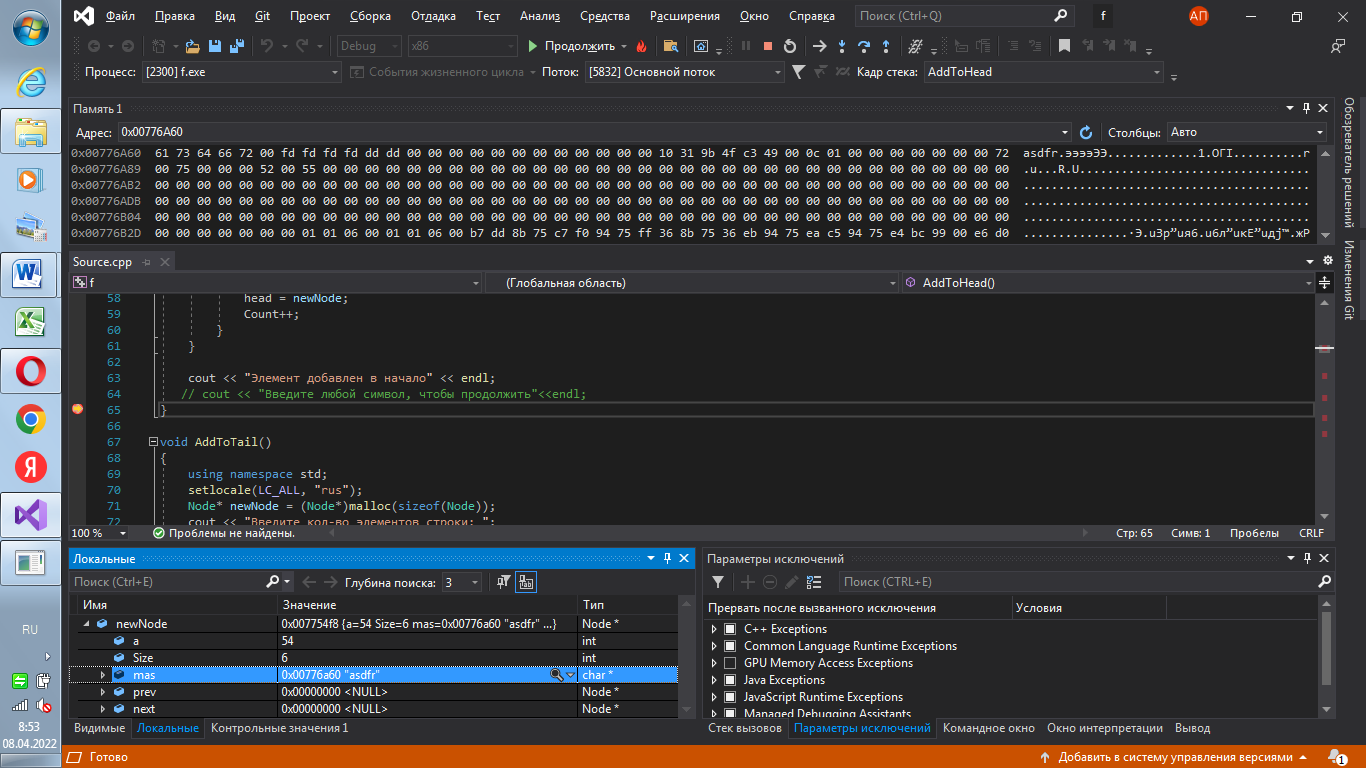
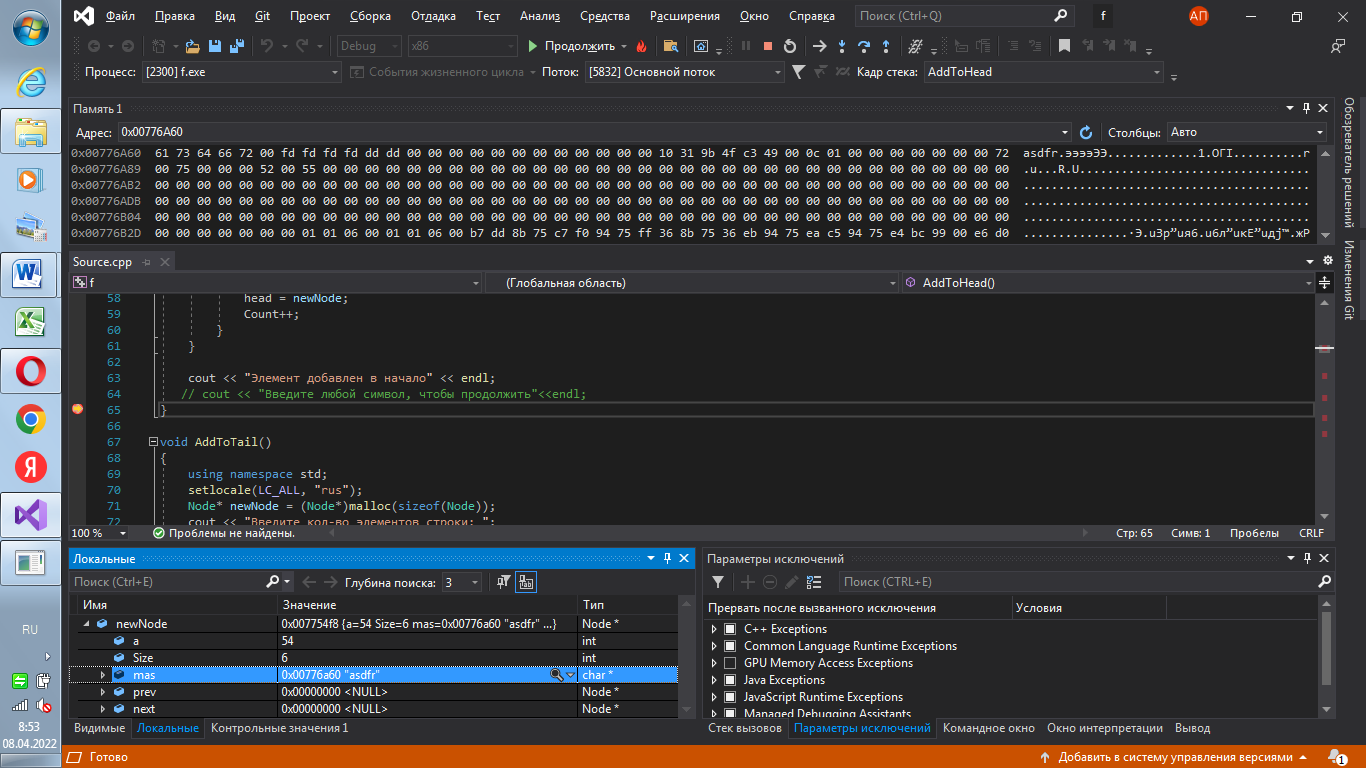
- длина строки

- указатель на строку

- указатель на предыдущий элемент

- указатель на следующий

- индикатор конца участка памяти

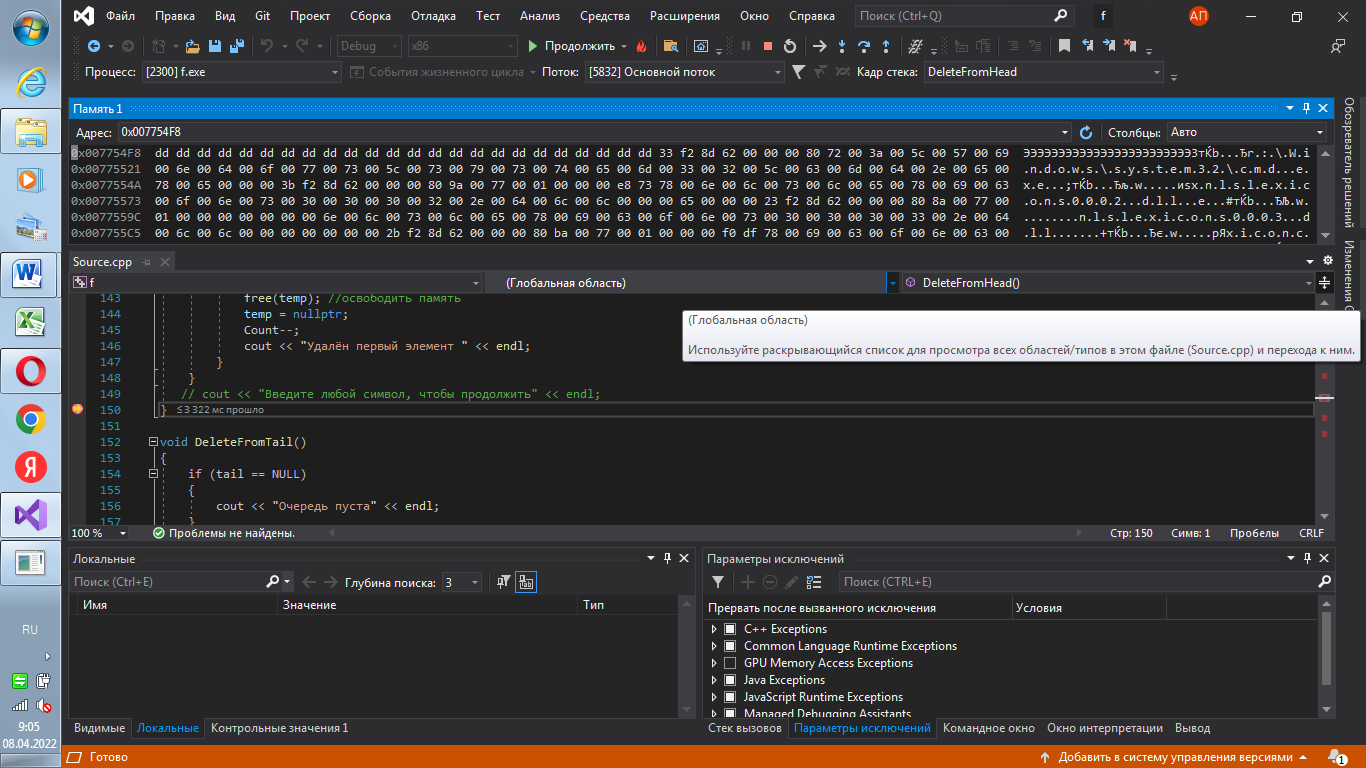
 

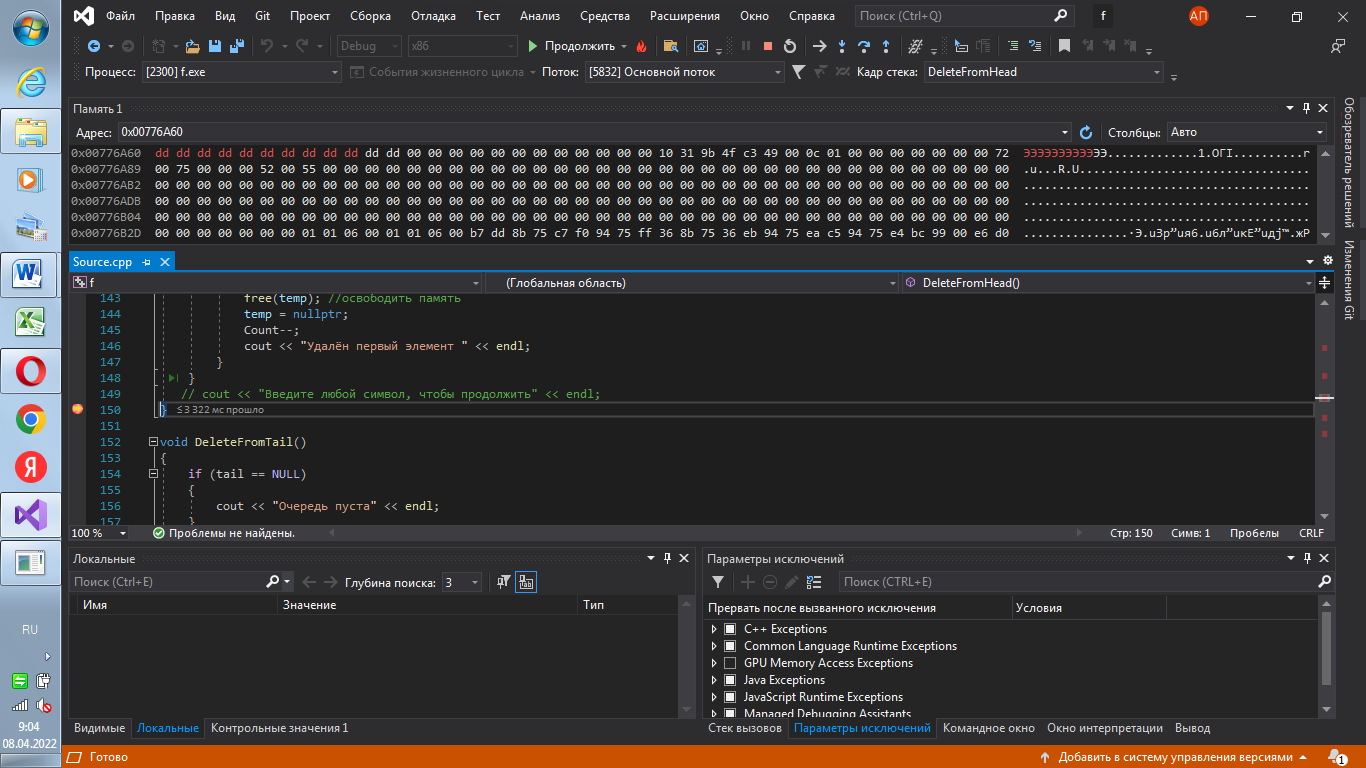
- элементы строки в 16СС

- индикатор конца участка памяти для строки

- элементы строки

**2 Удаление элемента**





Освобождение области памяти после удаления. Утечка памяти отсутствует.

**Листинг**

#include <iostream>

#include <conio.h>

using namespace std;

struct Node

{

int a;

int Size;

char\* mas;

Node\* prev;

Node\* next;

};

Node\* head = nullptr;

Node\* tail = nullptr;

int Count = 0;

void AddToHead()

{

using namespace std;

setlocale(LC\_ALL, "rus");

Node\* newNode = (Node\*)malloc(sizeof(Node));

cout << "Введите кол-во элементов строки: ";

while (!(cin >> newNode->Size) || (cin.peek() != '\n') || newNode->Size < 1 || newNode->Size > 80)

{

cin.clear();

while (cin.get() != '\n');

cout << "Некорректный ввод, введите число типа int от 1 до 80!" << endl;

cout << "Введите кол-во элементов массива: ";

}

newNode->mas = (char\*)malloc(newNode->Size \*sizeof(char)); //выделить память под массив

cout << "Введите строку: " << endl;

cin >> newNode->mas;

cout << "Введите целое число: ";

while (!(cin >> newNode->a) || (cin.peek() != '\n'))

{

cin.clear();

while (cin.get() != '\n');

cout << "Некорректный ввод, введите число типа int!" << endl;

cout << "Введите целое число: ";

}

if (newNode != nullptr)

{

if (head == nullptr)

{

head = tail = newNode;

//head->next = tail;

tail->next = nullptr;

head->prev = nullptr;

Count++;

}

else

{

newNode->next = head;

newNode->prev = nullptr;

head->prev = newNode;

head = newNode;

Count++;

}

}

cout << "Элемент добавлен в начало" << endl;

// cout << "Введите любой символ, чтобы продолжить"<<endl;

}

void AddToTail()

{

using namespace std;

setlocale(LC\_ALL, "rus");

Node\* newNode = (Node\*)malloc(sizeof(Node));

cout << "Введите кол-во элементов строки: ";

while (!(cin >> newNode->Size) || (cin.peek() != '\n') || newNode->Size < 1 || newNode->Size > 80)

{

cin.clear();

while (cin.get() != '\n');

cout << "Некорректный ввод, введите число типа int от 1 до 80!" << endl;

cout << "Введите кол-во элементов массива: ";

}

newNode->mas = (char\*)malloc(newNode->Size \* sizeof(char));

cout << "Введите строку: " << endl;

cin >> newNode->mas;

cout << "Введите целое число: ";

while (!(cin >> newNode->a) || (cin.peek() != '\n'))

{

cin.clear();

while (cin.get() != '\n');

cout << "Некорректный ввод, введите число типа int!" << endl;

cout << "Введите целое число: ";

}

if (newNode != nullptr)

{

if (tail == nullptr)

{

head = tail = newNode;

head->next = tail;

tail->next = nullptr;

tail->prev = head;

Count++;

}

else

{

newNode->next = nullptr;

newNode->prev = tail;

tail->next = newNode;

tail = newNode;

Count++;

}

}

cout << "Элемент добавлен в конец" << endl;

// cout << "Введите любой символ, чтобы продолжить" << endl;

}

void DeleteFromHead()

{

using namespace std;

setlocale(LC\_ALL, "rus");

if (head == NULL)

{

cout << "Очередь пуста" << endl;

}

else

{

if (Count == 1)

{

// delete[]head->mas;

free(head->mas);

free (head);

head = tail = nullptr;

Count--;

cout << "Удалён первый элемент" << endl;

}

else

{

Node\* temp = head;

head = head->next;

free (temp->mas);

//delete[]temp->mas; //освободить память динамического массива

temp->prev = nullptr;

temp->next = nullptr;

temp->mas = nullptr;

free(temp); //освободить память

temp = nullptr;

Count--;

cout << "Удалён первый элемент " << endl;

}

}

// cout << "Введите любой символ, чтобы продолжить" << endl;

}

void DeleteFromTail()

{

if (tail == NULL)

{

cout << "Очередь пуста" << endl;

}

else

{

if (Count == 1)

{

// delete []head->mas;

free(head->mas);

free(head);

head = tail = nullptr;

Count--;

cout << "Удалён последний элемент" << endl;

}

else

{

Node\* temp = tail;

tail = tail->prev;

free(temp->mas);

// delete []temp->mas; //освободить память динамического массива

temp->next = nullptr;

temp->prev = nullptr;

free(temp); //освободить память

Count--;

cout << "Удалён последний элемент" << endl;

}

}

// cout << "Введите любой символ, чтобы продолжить" << endl;

}

void Show()

{

using namespace std;

setlocale(LC\_ALL, "rus");

if (head == NULL)

{

cout << "Очередь пуста" << endl;

}

else

{

Node\* temp = head;

int j = 0;

do

{

cout << "Элемент " << j << ": ";

for (int i = 0; i < temp->Size; i++)

{

cout <<temp->mas[i];

}

cout << " a=" << temp->a;

cout << endl;

temp = temp->next;

j++;

} while (temp != NULL);

}

}

void Clear()

{

using namespace std;

setlocale(LC\_ALL, "rus"); //поддержка русских символов

Node\* temp;

if (head == nullptr)

{

cout << "Очередь пуста" << endl;

}

else

{

while (head != nullptr)

{

temp = head;

head = head->next;

// delete[] temp->mas;

temp->next = nullptr;

free(temp); //освободить память

Count--;

}

if (head == nullptr)

{

tail = nullptr;

cout << "Очередь очищена" << endl;

}

}

}

void PrintMenu() //меню

{

using namespace std;

setlocale(LC\_ALL, "rus"); //поддержка русских символов

system("cls");

cout << "1 - Добавить в начало" << endl;

cout << "2 - Добавить в конец" << endl;

cout << "3 - Удалить первый элемент" << endl;

cout << "4 - Удалить последний элемент" << endl;

// cout << "5 - Очистить" << endl;

// cout << "Показать элементы дека" << endl;

cout << "ESC.Выход" << endl;

cout << endl;

}

int main()

{

setlocale(LC\_ALL, "rus"); //поддержка русских символов

char ch;

do

{

PrintMenu();

ch = \_getch(); //считать нажатую клавишу

switch (ch)

{

case 49:

{

// PrintMenu();

AddToHead();

break;

//break;

}

case 50:

{

//PrintMenu();

AddToTail();

break;

}

case 51:

{

// PrintMenu();

DeleteFromHead();

break;

}

case 52:

{

//PrintMenu();

DeleteFromTail();

break;

}

// case 53:

// {

// //PrintMenu();

// Clear();

// break;

// }

case 54:

{

//PrintMenu();

Show();

break;

}

case 27:

{

break;

}

default:

//PrintMenu();

cerr << "Некорректный выбор пункта" << endl;

}

Show();

cout << endl;

cout << "Введите любой символ, чтобы продолжить" << endl;

ch = \_getch(); //считать нажатую клавишу

} while (ch != 27);

return 0;

}

**Вывод:** В ходе выполнения лабораторной памяти была изучена структура данных на динамической памяти – двусвязный список. Изучена структура элемента памяти списка.