МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

##### «МИНСКИЙ ГОСУДАРСТВЕННЫЙ КОЛЛЕДЖ ЦИФРОВЫХ ТЕХНОЛОГИЙ»

##### Группа 82ТП

## Отчет

## по лабораторной работе №26

"Разработка приложений с использованием списковых структур"

учебный предмет

«Основы алгоритмизации и программирования»

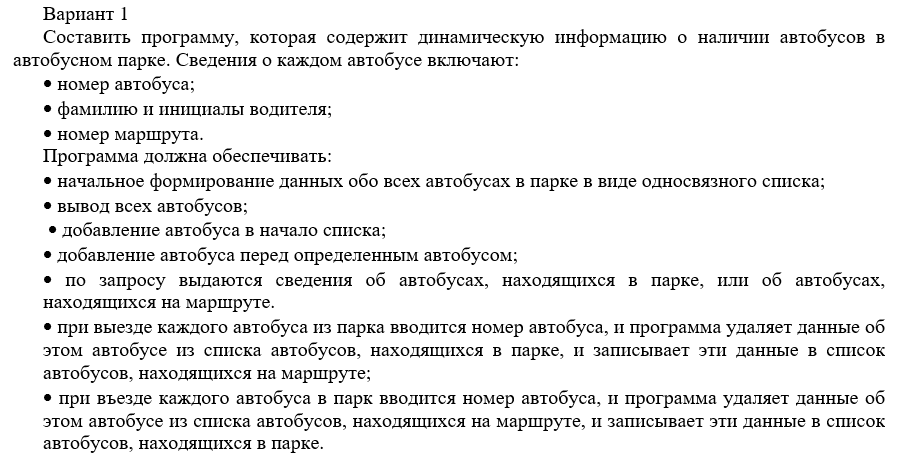
Исполнитель: Шатуха А. К.

Руководитель: Федкевич Д.А.

Минск 2025

Цель работы: сформировать умения разрабатывать приложения с использованием списковых структур.

Оснащение рабочего места: ПК, VS, методические указания для проведения лабораторных работ, индивидуальные задания на ЛР.

Задание 1:

Код:

#include <iostream>

#include <string>

using namespace std;

//Структура, описывающая данные об автобусе

struct Bus

{

int busNumber; //Номер автобуса

string driver; //Фамилия и инициалы водителя

string routeNumber; //Номер маршрута, теперь строка, позволяющая вводить буквы и числа

Bus\* next; //Указатель на следующий элемент односвязного списка

//Конструктор по умолчанию, который задаёт значения полей по умолчанию

//Это позволяет избежать ошибок, связанных с неинициализированными переменными

Bus() : busNumber(0), driver(""), routeNumber(""), next(nullptr) {}

};

//Функция для создания нового узла (объекта Bus)

Bus\* createBusNode()

{

Bus\* newBus = new Bus; //Выделяем память для нового автобуса, поля уже инициализированы конструктором

cout << "Введите номер автобуса: ";

cin >> newBus->busNumber; //Считываем номер автобуса

cin.ignore(); //Очистка буфера ввода (удаляем символ новой строки)

cout << "Введите фамилию и инициалы водителя: ";

getline(cin, newBus->driver); //Считываем фамилию и инициалы водителя, позволяя вводить пробелы

cout << "Введите номер маршрута (может содержать буквы): ";

getline(cin, newBus->routeNumber); //Считываем номер маршрута как строку

//Поле next уже установлено в nullptr в конструкторе

return newBus;

}

//Функция для начального формирования списка автобусов, находящихся в парке

void createInitialParkList(Bus\*& head)

{

head = nullptr; //Изначально список пуст

int n;

cout << "Введите количество автобусов в парке: ";

cin >> n; //Считываем количество автобусов

cin.ignore(); //Очистка буфера ввода

//Цикл для ввода данных каждого автобуса

for (int i = 0; i < n; i++)

{

cout << "\nАвтобус " << i + 1 << ":" << endl;

Bus\* newBus = createBusNode(); //Создаём новый узел с данными автобуса

//Если список пуст, данный автобус становится первым элементом

if (head == nullptr)

{

head = newBus;

}

else

{

//Иначе проходим по списку до его конца и добавляем новый узел

Bus\* temp = head;

while (temp->next != nullptr)

{

temp = temp->next;

}

temp->next = newBus;

}

}

}

//Функция для вывода информации обо всех автобусах из списка

void printBusList(Bus\* head)

{

//Если список пуст, выводим сообщение и завершаем функцию

if (!head)

{

cout << "Список пуст." << endl;

return;

}

//Проходим по всему списку, выводя данные каждого автобуса

Bus\* temp = head;

while (temp != nullptr)

{

cout << "Номер автобуса: " << temp->busNumber << endl;

cout << "Водитель: " << temp->driver << endl;

cout << "Номер маршрута: " << temp->routeNumber << endl;

cout << "-----------------------------" << endl;

temp = temp->next; //Переходим к следующему узлу

}

}

//Функция для добавления нового автобуса в начало списка

void addBusAtBeginning(Bus\*& head)

{

Bus\* newBus = createBusNode(); //Создаём новый узел с данными автобуса

newBus->next = head; //Привязываем новый узел к текущей голове списка

head = newBus; //Новый узел становится головой списка

cout << "Автобус добавлен в начало списка." << endl;

}

//Функция для добавления нового автобуса перед автобусом с определённым номером

void addBusBefore(Bus\*& head, int targetBusNumber)

{

Bus\* newBus = createBusNode();

//Если список пуст, то новый автобус станет первым элементом

if (head == nullptr)

{

cout << "Список пуст. Новый автобус будет добавлен как первый элемент." << endl;

head = newBus;

return;

}

//Если первый элемент уже имеет искомый номер, то добавляем новый узел в начало

if (head->busNumber == targetBusNumber)

{

newBus->next = head;

head = newBus;

cout << "Автобус добавлен перед автобусом с номером " << targetBusNumber << "." << endl;

return;

}

//Поиск узла, перед которым необходимо добавить новый автобус

Bus\* prev = nullptr;

Bus\* curr = head;

while (curr != nullptr && curr->busNumber != targetBusNumber)

{

prev = curr;

curr = curr->next;

}

//Если автобус с заданным номером не найден, выводим сообщение и освобождаем память нового узла

if (curr == nullptr)

{

cout << "Автобус с номером " << targetBusNumber << " не найден. Добавление не выполнено." << endl;

delete newBus;

return;

}

//Вставляем новый узел между prev и curr

prev->next = newBus;

newBus->next = curr;

cout << "Автобус добавлен перед автобусом с номером " << targetBusNumber << "." << endl;

}

//Функция для удаления автобуса из списка по его номеру

Bus\* removeBus(Bus\*& head, int targetBusNumber)

{

if (head == nullptr) return nullptr; //Если список пуст, возврат nullptr

Bus\* curr = head;

Bus\* prev = nullptr;

//Поиск нужного узла по номеру автобуса

while (curr != nullptr && curr->busNumber != targetBusNumber)

{

prev = curr;

curr = curr->next;

}

//Если узел не найден, возвращаем nullptr

if (curr == nullptr)

{

return nullptr;

}

//Если удаляется первый элемент списка, обновляем указатель head

if (prev == nullptr)

{

head = head->next;

}

else

{

//Иначе, переподключаем предыдущий элемент к следующему после удаляемого узла

prev->next = curr->next;

}

curr->next = nullptr; //Отсоединяем удалённый узел от списка

return curr; //Возвращаем указатель на удалённый узел

}

// Функция для перемещения автобуса из одного списка (source) в другой (destination)

void moveBus(Bus\*& source, Bus\*& destination, int targetBusNumber)

{

Bus\* busNode = removeBus(source, targetBusNumber); //Удаляем узел из исходного списка

if (busNode == nullptr)

{

cout << "Автобус с номером " << targetBusNumber << " не найден." << endl;

return;

}

//Добавляем удалённый узел в начало списка назначения

busNode->next = destination;

destination = busNode;

cout << "Автобус с номером " << targetBusNumber << " успешно перемещен." << endl;

}

//Функция для освобождения памяти, занятой списком

//Проходит по всему списку, удаляя каждый узел

void freeList(Bus\*& head)

{

while (head)

{

Bus\* temp = head; //Временный указатель на текущий узел

head = head->next; //Переходим к следующему узлу

delete temp; //Освобождаем память текущего узла

}

}

int main()

{

setlocale(LC\_ALL, "ru");

//Объявляем два указателя для поддержки двух списков:

//parkList - автобусы, находящиеся в парке

//routeList - автобусы, находящиеся на маршруте (выезд из парка)

Bus\* parkList = nullptr;

Bus\* routeList = nullptr;

int choice; //Переменная для выбора пункта меню

//Цикл, реализующий меню с выбором операций

do

{

// Выводим меню для пользователя

cout << "\nМеню:\n";

cout << "1. Сформировать начальный список автобусов в парке\n";

cout << "2. Вывести автобусы, находящиеся в парке\n";

cout << "3. Вывести автобусы, находящиеся на маршруте\n";

cout << "4. Добавить автобус в начало списка автобусов в парке\n";

cout << "5. Добавить автобус перед определенным автобусом в парке\n";

cout << "6. Автобус выезжает из парка (переместить из парка в маршрут)\n";

cout << "7. Автобус въезжает в парк (переместить из маршрута в парк)\n";

cout << "8. Выход\n";

cout << "Выберите действие: ";

cin >> choice; //Считываем выбор пользователя

cin.ignore(); //Очищаем буфер после ввода числа

switch (choice)

{

//Пункт 1 - Формирование начального списка автобусов в парке

case 1:

createInitialParkList(parkList);

break;

//Пункт 2 - Вывод автобусов, находящихся в парке

case 2:

cout << "\nАвтобусы в парке:\n";

printBusList(parkList);

break;

//Пункт 3 - Вывод автобусов, находящихся на маршруте

case 3:

cout << "\nАвтобусы на маршруте:\n";

printBusList(routeList);

break;

//Пункт 4 - Добавление автобуса в начало списка автобусов в парке

case 4:

addBusAtBeginning(parkList);

break;

//Пункт 5 - Добавление нового автобуса перед определенным автобусом в парке

case 5:

{

int targetNumber;

cout << "Введите номер автобуса, перед которым нужно добавить новый: ";

cin >> targetNumber;

cin.ignore();

addBusBefore(parkList, targetNumber);

break;

}

//Пункт 6 - Перемещение автобуса из парка на маршрут

case 6:

{

int busNum;

cout << "Введите номер автобуса, который выезжает из парка: ";

cin >> busNum;

cin.ignore();

moveBus(parkList, routeList, busNum);

break;

}

//Пункт 7 - Перемещение автобуса с маршрута обратно в парк

case 7:

{

int busNum;

cout << "Введите номер автобуса, который въезжает в парк: ";

cin >> busNum;

cin.ignore();

moveBus(routeList, parkList, busNum);

break;

}

case 8:

cout << "Выход из программы." << endl;

break;

default:

cout << "Неверный выбор. Попробуйте снова." << endl;

break;

}

}

while (choice != 8);

//Очистка выделенной памяти для обоих списков перед завершением программы

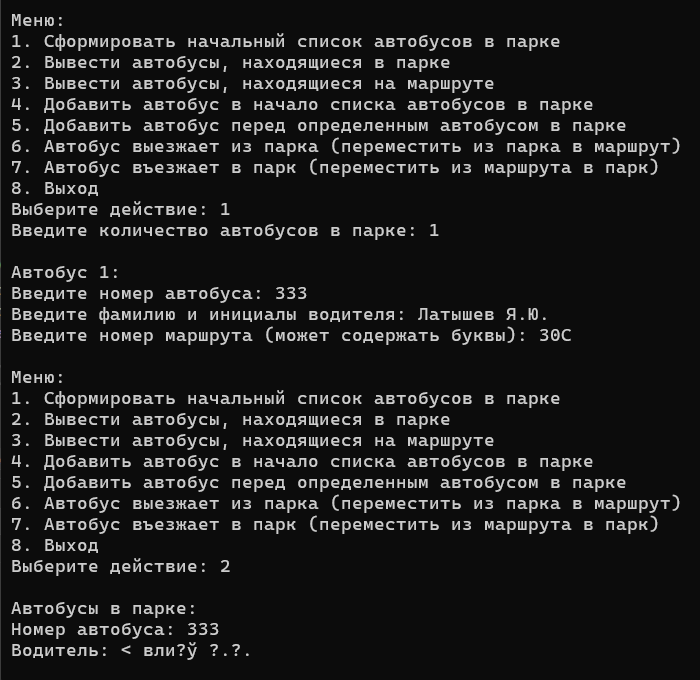
freeList(parkList);

freeList(routeList);

return 0;

}

Результат работы программы (копия экрана):



Ответы на контрольные вопросы:

1) Дать определение понятию «Список»

**Список** – это абстрактный тип данных, представляющий собой упорядоченную коллекцию элементов. В списке каждый элемент имеет своё место, причём порядок элементов имеет значение.

2) Проклассифицировать списки

**1. По способу хранения:**

* **Последовательные списки (массивы):** Элементы хранятся в непрерывном блоге памяти. Пример – статический массив или вектор (std::vector), обеспечивающий доступ по индексу.
* **Связные списки:** Каждый элемент (узел) хранится в отдельном блоке динамической памяти, а последовательность задаётся указателями, связывающими узлы. Примеры – односвязный список и двусвязный список.

**2. По типу связей между элементами:**

**Односвязные списки:** Каждый узел содержит данные и указатель только на следующий элемент. Пример в стандартной библиотеке – std::forward\_list.

**Двусвязные списки:** Каждый узел содержит два указателя: на предыдущий и на следующий элемент. Пример – std::list в STL.

**Циклические списки:** Последний элемент списка указывает на первый, образуя циклическую структуру. Такие списки могут быть как односвязными, так и двусвязными.

3) Каких видов бывают списки?

**Последовательный список (Динамический массив):** Элементы располагаются в непрерывном участке памяти. Обеспечивает быстрый доступ по индексу, но вставка или удаление элемента в середине может требовать сдвига элементов.

**Односвязный список:** Каждый узел содержит данные и указатель на следующий узел. Несмотря на то, что прямой доступ по индексу невозможен (для доступа требуется последовательный обход), вставка и удаление узлов производится эффективно без необходимости перемещать остальные элементы.

**Двусвязный список:** Каждый узел содержит два указателя – на предыдущий и на следующий узел. Это позволяет легко перемещаться по списку в обоих направлениях, а также — эффективно вставлять и удалять узлы в любом месте списка.

**Циклический список:** Особенность такого списка в том, что последний элемент указывает не на nullptr, а на первый элемент. Это удобно в ситуациях, когда требуется циклический обход элементов (например, для реализации очереди или кругового буфера).