Пермский национальный исследовательский политехнический университет

Электротехнический факультет

Информатика и вычислительная техника

ОТЧЕТ

О выполнении лабораторной работы №11
«Информационные динамические структуры»

Студент: Шадрин И. Д.
Группа: ИВТ-23-1Б
Преподаватель: Яруллин Д. В.

Пермь – 2024

1. Формулировка индивидуального задания

Написать программу, в которой создаются динамические структуры и выполнить их обработку в соответствии со своим вариантом.

Для каждого вариант разработать следующие функции:

1. Создание списка.
2. Добавление элемента в список (в соответствии со своим вариантом).
3. Удаление элемента из списка (в соответствии со своим вариантом).
4. Печать списка.
5. Запись списка в файл.
6. Уничтожение списка.
7. Восстановление списка из файла.

Записи в линейном списке содержат ключевое поле типа int. Сформировать однонаправленный список. Удалить из него элемент с заданным номером, добавить К элементов, начиная с заданного номера;

2. Описание использованных типов данных

При выполнении данной лабораторной работы использовался встроенный тип данных int, предназначенный для работы с целыми числами и класс string.

3. Исходные коды разработанных программ

Листинг 1: Исходные коды программы `main.o` (файл: `main.cpp`)

```
#include "list.h"

void init_with_rand_numbers(List *list, int n);

int main() {
    srand(time(nullptr));

    List *list = init_list();
    int n = rand() % 9 + 2;
    init_with_rand_numbers(list, n);
    print_list(list);

    int n1 = rand() % n;
    std::cout << "Delete " << n1 << " element" << std::endl;
    delete_by_index(list, n1);
    print_list(list);

    n1 = rand() % (n - 1);
    int k = rand() % 4 + 2;
    std::cout << "Add " << k << " elements after " << n1 << " element" << std::endl;

    for (int i = 0; i < k; ++i) {
        add_by_index(list, n1, rand() % 100);
        ++n1;
    }
    print_list(list);

    std::cout << "Input file name: ";
    std::string filename;
    std::getline(std::cin, filename);
    write_list_to_file(list, filename);

    delete_list(list);
    list = init_list();
    print_list(list);

    read_list_from_file(list, filename);
    print_list(list);

    delete_list(list);

    return 0;
}

void init_with_rand_numbers(List *list, int n) {
    for (int i = 0; i < n; ++i) {
        push_back(list, rand() % 100);
    }
}
```

Листинг 2: Исходные коды программы `main.o` (файл: `list.cpp`)

```
#include "list.h"

#include <fstream>

typedef struct Node {
    int data;
    Node* next;
} Node;

struct List {
    Node* head;
    Node* tail;
};

List *init_list() {
    List *list = (List *) calloc(1, sizeof(List));
    list->head = NULL;
    list->tail = NULL;
    return list;
}

bool is_empty(List *list) {
    return list->head == NULL;
}

void push_back(List *list, int elem) {
    Node *node = (Node *) calloc(1, sizeof(Node));
    node->data = elem;
    node->next = NULL;

    if (is_empty(list)) {
        list->head = node;
        list->tail = node;

        return;
    }

    list->tail->next = node;
    list->tail = node;
}

bool pop_back(List *list) {
    if (is_empty(list)) {
        return false;
    }

    Node *ptr = list->head, *prev_ptr = NULL;

    while (ptr->next) {
        prev_ptr = ptr;
        ptr = ptr->next;
    }
```

```
    if (ptr == list->head) {
        list->head = NULL;
        list->tail = NULL;
    }

    else {
        list->tail = prev_ptr;
        prev_ptr->next = NULL;
        free(ptr);
    }

    return true;
}

void print_list(List *list) {
    if (is_empty(list)) {
        std::cout << "List is empty" << std::endl;
        return;
    }

    Node *ptr = list->head;

    while (ptr) {
        std::cout << ptr->data << " ";
        ptr = ptr->next;
    }
    std::cout << std::endl;
}

void delete_list(List *list) {
    Node *ptr = list->head, *prev_ptr = NULL;

    while (ptr) {
        prev_ptr = ptr;
        ptr = ptr->next;
        free(prev_ptr);
    }

    free(list);
}

void delete_by_index(List *list, int n) {
    int i = 0;
    Node *ptr = list->head, *prev_ptr = NULL;

    while (i < n) {
        prev_ptr = ptr;
        ptr = ptr->next;
        ++i;
    }

    if (ptr == list->tail) {
        list->tail = prev_ptr;
    }
```

```
    if (ptr == list->head) {
        list->head = ptr->next;
    }

    else {
        prev_ptr->next = ptr->next;
    }

    free(ptr);
}

void add_by_index(List *list, int n, int k) {
    Node *node = (Node *) calloc(1, sizeof(Node));
    node->data = k;

    if (is_empty(list)) {
        list->head = node;
        list->tail = node;
        node->next = NULL;

        return;
    }

    int i = 0;
    Node *ptr = list->head;

    while (i < n) {
        ptr = ptr->next;
        ++i;
    }

    node->next = ptr->next;
    ptr->next = node;

    if (ptr == list->tail) {
        list->tail = node;
    }
}

void write_list_to_file(List *list, std::string filename) {
    std::ofstream file(filename);

    Node *ptr = list->head;

    while (ptr) {
        file << ptr->data << " ";
        ptr = ptr->next;
    }

    file.close();
}

bool read_list_from_file(List *list, std::string filename) {
```

```
    std::ifstream file(filename);

    if (!file.is_open()) {
        return false;
    }

    int elem;
    while (file >> elem) {
        push_back(list, elem);
    }

    file.close();
    return true;
}
```

Листинг 3: Исходные коды программы `main.o` (файл: `list.h`)

```
#ifndef LIST_H
#define LIST_H

#include <iostream>

typedef struct List List;

List *init_list();
bool is_empty(List *list);
void push_back(List *list, int elem);
bool pop_back(List *list);
void print_list(List *list);
void delete_list(List *list);

void delete_by_index(List *list, int n);
void add_by_index(List *list, int n, int k);

void write_list_to_file(List *list, std::string filename);
bool read_list_from_file(List *list, std::string filename);

#endif
```

4. Описание тестовых примеров

Таблица 1: Тестовые примеры

Ввод	Ожидаемый вывод	Вывод
1 60 86 79 28 61 70 33 Delete 6 element Add 4 elements after 3 element Input file name: test.txt	1 60 86 79 28 61 33 1 60 86 79 19 96 30 72 28 61 33 List is empty 1 60 86 79 19 96 30 72 28 61 33	1 60 86 79 28 61 33 1 60 86 79 19 96 30 72 28 61 33 List is empty 1 60 86 79 19 96 30 72 28 61 33
7 28 30 18 37 Delete 2 element Add 2 elements after 0 element Input file name: test.txt	7 28 18 37 7 58 22 28 18 37 List is empty 7 58 22 28 18 37	7 28 18 37 7 58 22 28 18 37 List is empty 7 58 22 28 18 37
68 6 72 9 99 99 55 55 5 6 Delete 9 element Add 5 elements after 7 element Input file name: test.txt	68 6 72 9 99 99 55 55 5 68 6 72 9 99 99 55 55 19 7 12 30 24 5 List is empty 68 6 72 9 99 99 55 55 19 7 12 30 24 5	68 6 72 9 99 99 55 55 5 68 6 72 9 99 99 55 55 19 7 12 30 24 5 List is empty 68 6 72 9 99 99 55 55 19 7 12 30 24 5

5. Скриншоты

```
ivan@DESKTOP-GIL324P:/mnt/c/GitHub/Labs_PSTU/Sem_2/Green/11$ g++ -o main.o list.cpp main.cpp
ivan@DESKTOP-GIL324P:/mnt/c/GitHub/Labs_PSTU/Sem_2/Green/11$ valgrind ./main.o
==111== Memcheck, a memory error detector
==111== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==111== Using Valgrind-3.19.0 and LibVEX; rerun with -h for copyright info
==111== Command: ./main.o
==111==
1 60 86 79 28 61 70 33
Delete 6 element
1 60 86 79 28 61 33
Add 4 elements after 3 element
1 60 86 79 19 96 30 72 28 61 33
Input file name: test.txt
List is empty
1 60 86 79 19 96 30 72 28 61 33
==111==
==111== HEAP SUMMARY:
==111==     in use at exit: 0 bytes in 0 blocks
==111==   total heap usage: 32 allocs, 32 frees, 92,480 bytes allocated
==111==
==111== All heap blocks were freed -- no leaks are possible
==111==
==111== For lists of detected and suppressed errors, rerun with: -s
==111== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
ivan@DESKTOP-GIL324P:/mnt/c/GitHub/Labs_PSTU/Sem_2/Green/11$ valgrind ./main.o
==112== Memcheck, a memory error detector
==112== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==112== Using Valgrind-3.19.0 and LibVEX; rerun with -h for copyright info
==112== Command: ./main.o
==112==
7 28 30 18 37
Delete 2 element
7 28 18 37
Add 2 elements after 0 element
7 58 22 28 18 37
Input file name: test.txt
List is empty
7 58 22 28 18 37
==112==
==112== HEAP SUMMARY:
==112==     in use at exit: 0 bytes in 0 blocks
==112==   total heap usage: 22 allocs, 22 frees, 92,320 bytes allocated
==112==
==112== All heap blocks were freed -- no leaks are possible
==112==
==112== For lists of detected and suppressed errors, rerun with: -s
==112== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
ivan@DESKTOP-GIL324P:/mnt/c/GitHub/Labs_PSTU/Sem_2/Green/11$ valgrind ./main.o
==113== Memcheck, a memory error detector
==113== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==113== Using Valgrind-3.19.0 and LibVEX; rerun with -h for copyright info
==113== Command: ./main.o
==113==
68 6 72 9 99 99 55 55 5 6
Delete 9 element
68 6 72 9 99 99 55 55 5
Add 5 elements after 7 element
68 6 72 9 99 99 55 55 19 7 12 30 24 5
Input file name: test.txt
List is empty
68 6 72 9 99 99 55 55 19 7 12 30 24 5
==113==
==113== HEAP SUMMARY:
==113==     in use at exit: 0 bytes in 0 blocks
==113==   total heap usage: 38 allocs, 38 frees, 92,576 bytes allocated
==113==
```

Рис. 1: Сборка и запуск программы `main.o` с помощью `valgrind`