МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

“НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

Кафедра защиты информации

Лабораторная работа №2 по дисциплине «Технологии и методы программирования»

### Алгоритмы и структуры данных.

### Вариант 2.

Группа: АИ-02

Выполнил: Григорьев Д.А.

Проверил: Медведев М.А.

Оценка:

Новосибирск 2022

**Цели и задачи работы:** Рассмотреть принципы построения простейших структур данных. Реализовать алгоритмы в рамках решения различных типовых задач.

**Задача.**

Первая часть: реализовать на языке СИ (инъекции плюсового синтаксиса не допускаются) структуры данных и их основные операции согласно варианту. Описать основную идею использования структур, которые ссылаются сами на себя.

Вторая часть: реализовать в коде заданные алгоритмы, используя стратегии построения алгоритмов.

**Задание**

Рассмотреть предметную область на основании изначальных данных, построить модели программного кода с использованием паттерна Singleton и SOLID. Реализовать задачу и выстроить аргументированную защиту/критику кода оппонента. Результатом работы программного кода должна быть имитация процесса задания.

**Вариант 2.**

Циклический linked list

Имя входного файла: стандартный ввод

Имя выходного файла: стандартный вывод

Ограничение по времени:2 секунды

Ограничение по памяти: 64 мегабайта

Изначально в кузове игрушечного автомобиля присутствует N песчинок, каждая из них имеет объем Vi и вес Ci, сами песчинки расщепить на отдельные детали невозможно. Так же существует лоток вместимостью V. Параметры: N, Vi, Ci – псевдорандомны и предопределены.

Требуется разместить набор различных песчинок максимального веса, и при этом объем песчинок не превышает объем лотка.

**Текст программы:**

***Task 1:***

#include <stdio.h>

#include <stdlib.h>

struct Node

{

int data;

struct Node\* next;

};

struct Node\* newNode(int data, struct Node\* nextNode)

{

struct Node\* node = (struct Node\*)malloc(sizeof(struct Node));

node->data = data;

node->next = nextNode;

return node;

}

struct Node\* constList(const int\* keys, int n)

{

struct Node \*head = NULL, \*node = NULL;

for (int i = n - 1; i >= 0; --i)

{

node = newNode(keys[i], node);

}

head = node;

return head;

}

//я не понимаю как это работает, но оно работает

void insertNode(struct Node\* head, int new\_data, int after)

{

struct Node\* new\_node = NULL;

struct Node\* tmp = head;

while (tmp) {

if (tmp->data == after) {

new\_node = (struct Node\*)malloc(sizeof(struct Node));

if (new\_node == NULL) {

printf("Failed to insert element");

}

new\_node->data = new\_data;

new\_node->next = tmp->next;

tmp->next = new\_node;

return;

}

tmp = tmp->next;

}

}

//это тоже непонятно как работает

void deleteNode(struct Node\* head, int value)

{

struct Node\* new\_node = NULL;

struct Node\* tmp = head;

while (tmp) {

if (tmp->data == value) {

new\_node = (struct Node\*)malloc(sizeof(struct Node));

if (new\_node == NULL) {

printf("Failed to delete element");

}

new\_node = tmp->next->next;

tmp->next = new\_node;

return;

}

tmp = tmp->next;

}

}

void printList(struct Node\* head)

{

struct Node\* ptr = head;

while (ptr)

{

printf("%d -> ", ptr->data);

ptr = ptr->next;

}

if (!ptr) {

ptr = head;

printf("%d <- first node of list\n", ptr->data);

}

}

int main(void)

{

const int keys[5] = { 1, 2, 3, 4, 5 };

//int n = sizeof(keys) / sizeof(keys[0]);

struct Node\* head = constList(keys, 5);

struct Node\* sub;

printList(head);

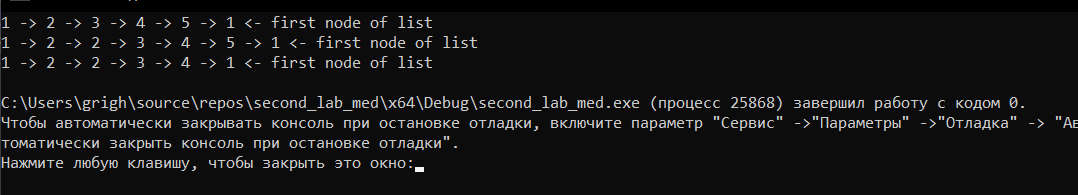
insertNode(head, 2, 2);

printList(head);

deleteNode(head, 4);

printList(head);

return 0;

}

**Анализ кода:**

Главная идея ссылки структуры на структуру – связанность и меньшие затраты по памяти.

Мы ссылаемся на структуру в структуре, чтобы мы могли сделать последовательность “точек”, и затем с ними работать – изменять, добавлять и удалять.

Теоретическая проблема может быть только к обращению к нулевому указателю, но решается легко проверкой на NULL.

**Task 2:**

//Изначально в кузове игрушечного автомобиля присутствует N песчинок, каждая из них имеет объем Vi и вес Ci,

//сами песчинки расщепить на отдельные детали невозможно.Так же существует лоток вместимостью V.

//Параметры: N, Vi, Ci – псевдорандомны и предопределены.

//

//Требуется разместить набор различных песчинок максимального веса, и при этом объем песчинок не превышает объем лотка.

//пеебором не вариант, нашел объяснение решения на хабре

#include <iostream>

int main(void)

{

const int k = 3; // количество предметов

const double W = 4; // вместимость ковша

double w[k] = { 4, 1, 3 }; // масса

double p[k] = { 4000, 2500, 2000 }; // стоимость

double mass[4][k]; //массив для расчетов

double weight = 0, volume = 0; //суммарный объём // суммарный вес

for (int i = 0; i < k; ++i) //подсчет коэффициентов и заполнение строк массива

{

mass[0][i] = (p[i] / w[i]);

mass[1][i] = w[i];

mass[2][i] = p[i];

mass[3][i] = i + 1;

}

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

{

for (int j = 0; j < k; ++j)

{

double buffer;

if (mass[0][i] > mass[0][j])

{

for (int z = 0; z < 4; ++z)

{

buffer = mass[z][i];

mass[z][i] = mass[z][j];

mass[z][j] = buffer;

}

}

}

}

std::cout << "Goods: ";

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

{

weight += mass[1][i];

if (weight <= W)

{

std::cout << mass[3][i] << " ";

volume += mass[2][i];

}

else

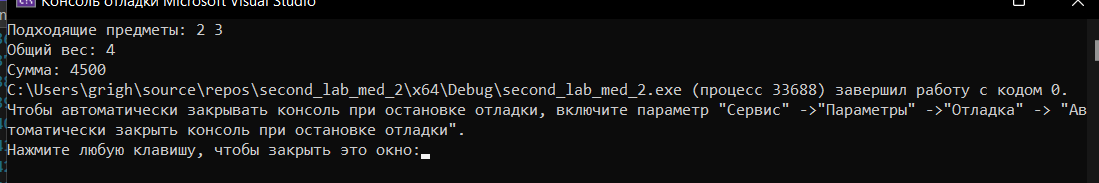
{

weight -= mass[1][i];

}

}

std::cout << "\nTotal weight: " << weight << std::endl << "Amount: " << pay;

}****

**Анализ кода:**

Задача является вводной в теорию сложности алгоритмов, сама по себе она класса NP-полных задач. Для решения используется жадный алгоритм Радо — Эдмондса (жадный алгоритм поиска базы минимального веса), нам необходимо отсортировать вещи по их ценности и затем поместить в рюкзак предметы с наибольшей ценностью. Нам необходимо построить множество, из других подмножеств. Мат. модель будет доказательством теоремы этого алгоритма.

sort(X)

B←∅

for i←0 to n−1

if B∪X[i]∈I

B←B∪X[i]

**Выводы**

Провел исследовательскую работу по изучению различных алгоритмов. Немного разобрался с динамическим программированием, а так же узнал про NP и P классы задач. Так-же попытался немного разобрать теоретический материал.