**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ СТРОИТЕЛЬНЫЙ УНИВЕРСИТЕТ»**

Институт цифровых технологий и моделирования в строительстве

Кафедра информатики и прикладной математики

**КУРСОВАЯ РАБОТА**

по дисциплине

«\_\_\_\_\_\_\_\_Алгоритмизация и программирование \_\_\_\_\_\_\_\_\_\_\_»

Тема:

«\_\_\_\_\_\_\_\_Разработка информационно-поисковой системы с использованием графических средств С/С++\_\_\_\_»

|  |  |
| --- | --- |
| Выполнил обучающийся | ИЦТМС 1-6, Сигаев Иван Романович |
|  | (институт (филиал), курс, группа, Ф.И.О.) |
| Руководитель курсового(й) проекта (работы) | доц., к.т.н., профессор, Зоткин С.П. |
|  | (ученое звание, ученая степень, должность, Ф.И.О.) |
| К защите |  |
|  | (дата, подпись руководителя) |
| Курсовой(ая) проект (работа) защищен(-а) с оценкой |  |
|  | (оценка цифрой и прописью) |
| Руководитель курсового(й) проекта (работы) |  |
|  | (дата, подпись руководителя) |
| Председатель аттестационной комиссии |  |
|  | (ученое звание, ученая степень, должность, Ф.И.О.) |
| Члены комиссии: |  |
|  |  |
|  |  |
|  | (дата, подпись члена комиссии) |
|  | |

г. Москва

2024г.

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ СТРОИТЕЛЬНЫЙ УНИВЕРСИТЕТ»**

Институт \_\_\_\_ цифровых технологий и моделирования в строительстве \_\_\_\_\_

Кафедра \_\_\_\_\_\_ информатики и прикладной математики \_\_\_\_\_\_\_\_\_

Дисциплина*\_\_\_\_* Алгоритмизация и программирование *\_\_\_\_*

**ЗАДАНИЕ**

**НА ВЫПОЛНЕНИЕ КУРСОВОГО ПРОЕКТИРОВАНИЯ (КУРСОВОЙ РАБОТЫ)**

ФИО обучающегося: Сигаев Иван Романович

Курс, группа: ИЦТМС 1-6

1. Тема проекта (работы) «Разработка информационно-поисковой системы с использованием графических средств С/С++»: список заказов сервиса доставки еды
2. Исходные данные к курсовому(й) проекту (работе): адрес заказа, тип продукта, дата поступления заказа, вес заказа, цена заказа (содержатся в файле orders.dat)
3. Содержание текстовой части (перечень подлежащих разработке вопросов):

* Каков адрес наиболее дорогого заказа?
* Какой заказ был сделан последним?
* Алфавитный список заказов по типу продукта.
* Обратный алфавитный список заказов по типу продукта.
* Список заказов на сумму свыше 1000 руб.
* Каково количество заказов пиццы?
* Диаграмма - процентное соотношение цен заказов на продукт.
* Разные продукты с совпадающей ценой (первый).
* Разные продукты с совпадающей ценой (список).

1. Перечень графического и иного материала (с точным указанием обязательных чертежей)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

График выполнения курсового(й) проекта (работы):

|  |  |  |  |
| --- | --- | --- | --- |
| № | Наименование этапа выполнения курсового(й) проекта (работы) | Срок выполнения | Процент выполнения  курсового(й) проекта (работы) |
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |

1. Дата выдачи задания \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель курсового(й) проекта (работы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

**1.Введение .................................................................................................................... 5**

1.1 Исходные данные ................................................................................................... 5

1.2 Демонстрация работы функций ............................................................................ 6

1.2.1. Функция поиска максимума (самый дорогой заказ) ....................................... 7

1.2.2. Функция поиска максимума (самый последний заказ) ................................... 7

1.2.3. Прямой алфавитный список заказов ................................................................ 7

1.2.4. Обратный алфавитный список заказов ............................................................. 7

1.2.5. Сложный вопрос (поиск совпадения) ................................................................ 7

1.2.6. Сложный вопрос, список (поиск совпадения) .................................................. 8

1.2.7. Диаграмма (соотношение прибыльности товаров) .......................................... 8

**2 Программный код .................................................................................................... 8**

2.1. Спецификация ....................................................................................................... 21

**3 Заключение ............................................................................................................... 21**

3.1 Инструкция пользователя ...................................................................................... 22

**4 Список литературы.................................................................................................. 26**

1. Введение

Программа представляет собой информационно-поисковую систему для учета заказов сервиса доставки еды. С помощью функций сортировки и поиска осуществлено более удобное получение информации для пользователя.

Программа выводит полный список исходных данных и позволяет:

* Узнать адрес самого дорогого заказа.
* Просмотреть последний из заказов.
* Вывести алфавитный и обратный алфавитный списки прибыли от разных категорий продуктов.
* Вывести список всех заказов на сумму более 1000 руб.
* Узнать количество заказов пиццы.
* Составить диаграмму, показывающую процент доходности различных категорий продуктов.
* Найти разные продукты с совпадающей ценой (список и первое совпадение).

В программе осуществлена навигация по двухуровневому меню, возможность работы с обоими уровнями.

Код программы написан в среде программирования Microsoft Visual Studio на ЯВУ C++ в консольном варианте и в виде приложения Windows Forms.

В данном отчете представлен консольный вариант приложения.

1.1. Исходные данные

Исходные данные хранятся в файле “orders.dat”, который содержит 15 записей.

Одна запись содержит 5 полей:

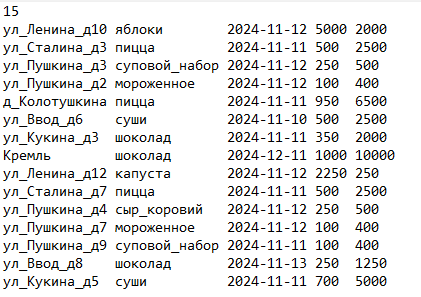
Строковые: Адрес заказа, тип продукта, дата заказа.

Числовые: Вес заказа, цена заказа.

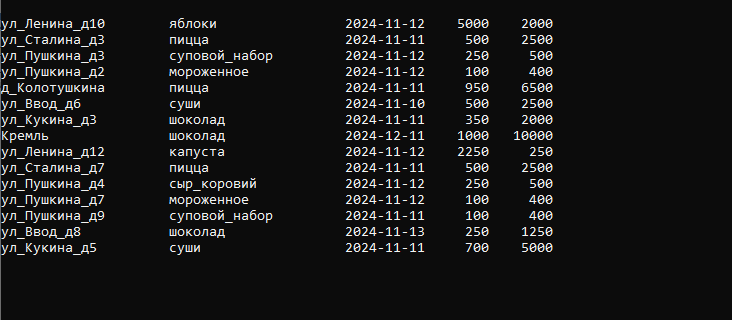
Первая строка указывает на общее количество записей, это сделано для

возможности контроля целостности данных.

Содержание файла с исходными данными:



Вид данных, выведенных из исходного файла, в приложении:

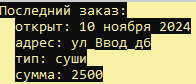


1.2. Демонстрация работы функций:

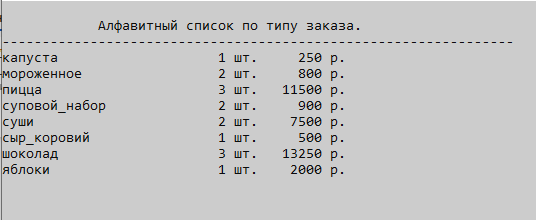
1.2.1. Функция поиска максимума (самый дорогой заказ):



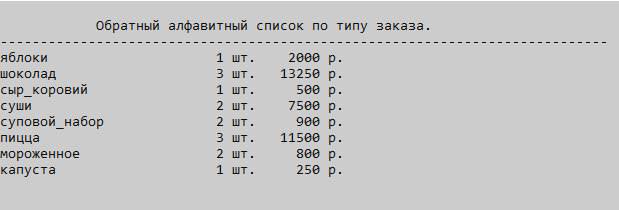
1.2.2. Функция поиска максимума (самый последний заказ):



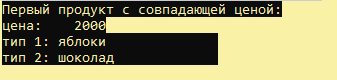
1.2.3. Прямой алфавитный список заказов:



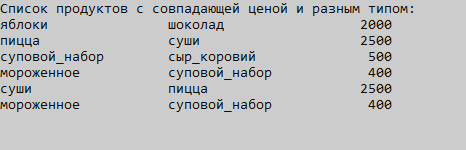
1.2.4. Обратный алфавитный список:



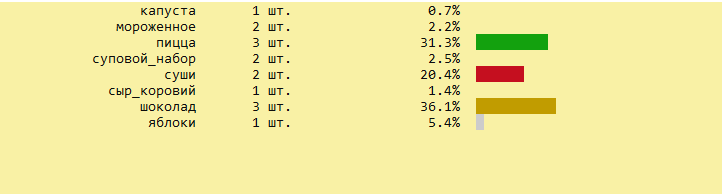
1.2.5. Сложный вопрос (поиск совпадения):



1.2.6. Сложный вопрос, список (поиск совпадения):



1.2.7. Диаграмма (соотношение прибыльности товаров):



2. Программный код

#include "stdafx.h"

#include <cstdio>

#include <cstdlib>

#include <cstring>

#include <clocale>

#include <conio.h>

using namespace std;

using namespace System;

using namespace System::IO;

// Структура описывающая один заказ еды.

struct order

{

char address[20]; // Адрес заказа.

char kind[20]; // Тип продукта.

char date[11]; // Дата поступления заказа.

long weight; // Вес заказа.

long price; // Цена заказа.

};

// Двусторонний связный список, хранящий

// данные о суммарной цене и кол-ве

// заказов определенного типа.

struct list

{

char kind[20]; // Тип продукта

long count; // Количество заказов с продуктом.

long price; // Суммарная цена заказов.

struct list\* next; // Следующий узел списка.

struct list\* prev; // Предыдущий узел списказ.

};

unsigned int count; // Количество заказов еды.

struct order\* orders; // Массив заказов еды.

struct list\* sp; // Список заказов еды по типу.

#define ENTER 13

#define ESC 27

#define UP 72

#define DOWN 80

#define HOME 71

#define END 79

// Описание:

// Печатает в консоль (переписывает весь экран)

// данные массива orders.

void print\_orders()

{

Console::ForegroundColor = ConsoleColor::White;

Console::BackgroundColor = ConsoleColor::Black;

Console::Clear();

for (unsigned int i = 0; i < count; i++)

{

printf(

"\n%-20s %-20s %11s %7ld %7ld",

orders[i].address,

orders[i].kind,

orders[i].date,

orders[i].weight,

orders[i].price

);

}

(void)\_getch();

}

// Описание:

// Печатает в консоль (после курсора)

// заказ с наибольшей ценой.

void print\_most\_expensive\_order()

{

struct order\* best = &orders[0];

for (unsigned int i = 1; i < count; i++)

{

if (best->price < orders[i].price)

{

best = &orders[i];

}

}

Console::ForegroundColor = ConsoleColor::Yellow;

Console::BackgroundColor = ConsoleColor::Black;

Console::CursorLeft = 10;

printf("Самый дорогой заказ: %ld руб.\n", best->price);

Console::CursorLeft = 10;

printf("Его адрес: %s\n", best->address);

(void)\_getch();

}

// Описание:

// Переводит дату из формата 'гггг\_мм\_дд' (from)

// в формат 'дд месяц гггг' (to).

//

// Если участки памяти to и from пересекаются,

// то поведение функции неопределено.

//

// Аргументы:

// to -> Указатель на массив (24+ симв.),

// в котором будет сохранена дата в

// итоговом формате как

// нуль-терминированная строка.

// from -> Указатель на нуль-терминированную

// строку (11 симв. включая нуль),

// в которой содержится дата в

// изначальном формате.

void to\_pretty\_date(char\* to, const char\* from)

{

char tmp[3];

const char\* months[] = {

"января","февраля","марта","апреля","мая","июня",

"июля","августа","сентября","октября","ноября","декабря"

};

strcpy(to, from + 8);

strcat(to, " ");

strncpy(tmp, from + 5, 2);

tmp[2] = '\0';

strcat(to, months[atoi(tmp) - 1]);

strcat(to, " ");

strncat(to, from, 4);

}

// Описание:

// Печатает в консоль (после курсора)

// заказ с самой поздней датой.

//

// Если заказов с такой датой несколько,

// то печатает любой из них.

void last\_order()

{

struct order\* best = &orders[0];

for (unsigned int i = 0; i < count; i++)

{

if (strcmp(orders[i].date, best->date) < 0)

{

best = &orders[i];

}

}

char pretty\_date[17];

to\_pretty\_date(pretty\_date, best->date);

Console::ForegroundColor = ConsoleColor::Yellow;

Console::BackgroundColor = ConsoleColor::Black;

Console::CursorLeft = 10;

printf("Последний заказ:\n");

Console::CursorLeft = 12;

printf("открыт: %s\n", pretty\_date);

Console::CursorLeft = 12;

printf("адрес: % s\n", best->address);

Console::CursorLeft = 12;

printf("тип: %s\n", best->kind);

Console::CursorLeft = 12;

printf("сумма: %ld\n", best->price);

(void)\_getch();

}

// Описание:

// Печатает в консоль (после курсора)

// количество заказов пиццы.

void print\_pizza\_order\_count()

{

unsigned int total = 0;

for (unsigned int i = 0; i < count; i++)

{

if (strcmp(orders[i].kind, "пицца") == 0) total++;

}

Console::ForegroundColor = ConsoleColor::Yellow;

Console::BackgroundColor = ConsoleColor::Black;

Console::CursorLeft = 10;

printf("\nКоличество заказов пиццы: %u", total);

(void)\_getch();

}

// Описание:

// Выдкляет память и записывает новый узел списка,

// с пустыми указателями на следующий и предыдущий

// узлы.

//

// Аргументы:

// order -> Единственный заказ который будет

// описываться узлом.

//

// Возвращает:

// Новый узел списка. Чтобы освободить память

// можно использовать free().

struct list\* create\_list\_node(struct order order)

{

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

if (!node) abort();

strcpy(node->kind, order.kind);

node->count = 1;

node->price = order.price;

node->next = NULL;

node->prev = NULL;

return node;

}

// Описание:

// Добавляет заказ в список sp сортируя его по типу

// заказа, при надобности выделяя память.

//

// Функция гарантирует, что sp всегда будет

// указывать на начало списка.

//

// Если порядок сортировки списка sp был нарушен,

// то поведение функции неопределено.

//

// Аргументы:

// order -> Заказ, который будет добавлен в список sp.

void insert\_order(struct order order)

{

if (!sp)

{

sp = create\_list\_node(order);

return;

}

struct list\* node = sp;

while (true)

{

int cmp = strcmp(order.kind, node->kind);

if (cmp == 0)

{

node->count++;

node->price += order.price;

return;

}

else if (cmp > 0)

{

if (!node->next)

{

node->next = create\_list\_node(order);

node->next->prev = node;

return;

}

node = node->next;

}

else

{

struct list\* tmp = node->prev;

node->prev = create\_list\_node(order);

node->prev->next = node;

node->prev->prev = tmp;

if (tmp) tmp->next = node->prev;

if (sp->prev) sp = sp->prev;

return;

}

}

}

// Описание:

// Печатает в консоль (переписывает весь экран)

// данные списка sp в алфавитном порядке

// (сортировка идет по типу заказа).

void list\_alpha()

{

Console::ForegroundColor = ConsoleColor::Black;

Console::BackgroundColor = ConsoleColor::Gray;

Console::Clear();

printf("\n");

Console::CursorLeft = 12;

printf("Алфавитный список по типу заказа.");

printf("\n\r----------------------------------------------------------------");

for (struct list\* nt = sp; nt != NULL; nt = nt->next)

{

printf(

"%\n\r%-20s %7ld шт. %7ld р.",

nt->kind,

nt->count,

nt->price

);

}

(void)\_getch();

}

// Описание:

// Печатает в консоль (переписывает весь экран)

// данные списка sp в обратном алфавитном порядке

// (сортировка идет по типу заказа).

void list\_alpha\_reverse()

{

Console::ForegroundColor = ConsoleColor::Black;

Console::BackgroundColor = ConsoleColor::Gray;

Console::Clear();

printf("\n");

Console::CursorLeft = 12;

printf("Обратный алфавитный список по типу заказа.");

printf("\n\r----------------------------------------------------------------------------");

struct list\* tail = sp;

while (tail->next != NULL) tail = tail->next;

for (struct list\* nt = tail; nt != NULL; nt = nt->prev)

{

printf(

"%\n\r%-20s %7ld шт. %7ld р.",

nt->kind,

nt->count,

nt->price

);

}

(void)\_getch();

}

// Описание:

// Печатает в консоль (переписывает весь экран)

// список заказов из массива orders, цена

// которых превышает 1000.

void print\_expensive\_list()

{

Console::ForegroundColor = ConsoleColor::Black;

Console::BackgroundColor = ConsoleColor::Gray;

Console::Clear();

printf("\n");

Console::CursorLeft = 12;

printf("Список заказов на сумму более 1.000 руб");

printf("\n\r----------------------------------------------------------------");

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

{

if (orders[i].price > 1000)

{

printf(

"%\n\r%-20s %-20s %7ld г %7ld р.",

orders[i].address,

orders[i].kind,

orders[i].weight,

orders[i].price

);

}

}

(void)\_getch();

}

// Описание:

// Печатает в консоль (переписывает весь экран)

// диаграмму по наиболее дорогим заказам,

// основанную на данных списка sp.

void diagram()

{

Console::BackgroundColor = ConsoleColor::Yellow;

Console::Clear();

long total = 0;

for (unsigned int i = 0; i < count; i++)

{

total += orders[i].price;

}

ConsoleColor color = ConsoleColor::Black;

int ncolor = 0;

for (struct list\* nt = sp; nt != NULL; nt = nt->next, ncolor++, color++)

{

char str[20];

sprintf(str, "%3.1f%%", (nt->price \* 100.0 / total));

Console::ForegroundColor = ConsoleColor::Black;

Console::BackgroundColor = ConsoleColor::Yellow;

Console::CursorLeft = 5;

printf("%20s %7ld шт. %20s", nt->kind, nt->count, str);

Console::BackgroundColor = color;

Console::CursorLeft = 60;

printf("%\*.s\n", nt->price \* 30 / total, "");

if (ncolor == 13)

{

color = ConsoleColor::Black;

ncolor = 0;

}

}

(void)\_getch();

return;

}

// Описание:

// Печатает в консоль (после курсора)

// первое совпадение цен заказа для

// двух заказов из массива orders с

// разными типами.

void match\_one()

{

Console::ForegroundColor = ConsoleColor::Yellow;

Console::BackgroundColor = ConsoleColor::Black;

struct order a, b;

bool found\_match = false;

for (unsigned int i = 0; i < count; i++)

{

for (unsigned int j = i + 1; j < count; j++)

{

a = orders[i];

b = orders[j];

if (strcmp(a.kind, b.kind) == 0) continue;

if (a.price != b.price) continue;

found\_match = true;

goto finish;

}

}

finish:

Console::CursorLeft = 10;

printf("Первый продукт с совпадающей ценой:\n");

Console::CursorLeft = 10;

if (found\_match)

{

printf("цена: %7ld\n", a.price);

Console::CursorLeft = 10;

printf("тип 1: %-20s\n", a.kind);

Console::CursorLeft = 10;

printf("тип 2: %-20s\n", b.kind);

}

else printf("Совпадений не найдено.\n");

(void)\_getch();

}

// Описание:

// Печатает в консоль (переписывает весь экран)

// все совпадения цен заказа для

// двух заказов из массива orders с

// разными типами.

void match()

{

Console::ForegroundColor = ConsoleColor::Black;

Console::BackgroundColor = ConsoleColor::Gray;

Console::Clear();

printf("Список продуктов с совпадающей ценой и разным типом:");

for (unsigned int i = 0; i < count; i++)

{

for (unsigned int j = i + 1; j < count; j++)

{

struct order a = orders[i];

struct order b = orders[j];

if (strcmp(a.kind, b.kind) == 0) continue;

if (a.price != b.price) continue;

printf("\r\n%-20s %-20s %7ld", a.kind, b.kind, a.price);

}

}

(void)\_getch();

}

// Описание:

// Отрисовывает меню и позволяет выбрать

// одну из функций приложения при помощи

// клавиатуры.

void menu()

{

const char\* options[] = {

" ",

"Адрес самого дорогого заказа ",

"Последний заказ ",

"Алфавитный список по типу ",

"Алфавитный список по типу (обратный) ",

"Заказы на сумму свыше 1.000 р. ",

"Количество заказов пиццы ",

"Диаграмма ",

"Продукты с совпадающей ценой ",

"Продукты с совпадающей ценой (первый) ",

"Исходные данные ",

"Выход ",

" "

};

int options\_count = sizeof(options) / sizeof(options[0]);

int first\_option = 1;

int last\_option = options\_count - 2;

Console::BackgroundColor = ConsoleColor::Yellow;

Console::Clear();

Console::ForegroundColor = ConsoleColor::Black;

Console::BackgroundColor = ConsoleColor::Gray;

for (int i = 0; i < options\_count; i++)

{

Console::CursorLeft = 10;

Console::CursorTop = 4 + i;

printf(" %s ", options[i]);

}

int y = 1;

int last\_y = y;

while (1)

{

Console::ForegroundColor = ConsoleColor::Black;

Console::BackgroundColor = ConsoleColor::Gray;

Console::CursorLeft = 10 + 1;

Console::CursorTop = 4 + last\_y;

printf("%s", options[last\_y]);

Console::ForegroundColor = ConsoleColor::Black;

Console::BackgroundColor = ConsoleColor::Yellow;

Console::CursorLeft = 10 + 1;

Console::CursorTop = 4 + y;

printf("%s", options[y]);

Console::CursorTop = 4 + options\_count;

last\_y = y;

switch (\_getch())

{

case DOWN: y++; break;

case UP: y--; break;

case HOME: y = first\_option; break;

case END: y = last\_option; break;

case ENTER:

switch (y)

{

case 1: print\_most\_expensive\_order(); return;

case 2: last\_order(); return;

case 3: list\_alpha(); return;

case 4: list\_alpha\_reverse(); return;

case 5: print\_expensive\_list(); return;

case 6: print\_pizza\_order\_count(); return;

case 7: diagram(); return;

case 8: match(); return;

case 9: match\_one(); return;

case 10: print\_orders(); return;

case 11: exit(0);

}

case ESC: exit(0);

}

if (y > last\_option) y = first\_option;

if (y < first\_option) y = last\_option;

}

}

int main(array<System::String^>^ args)

{

setlocale(LC\_CTYPE, "Russian");

Console::CursorVisible::set(false);

Console::BufferHeight = Console::WindowHeight;

Console::BufferWidth = Console::WindowWidth;

const char\* path = "orders.dat";

FILE\* file = fopen(path, "r");

if (file == NULL)

{

printf(

"\nошибка: Не удается открыть файл %s "

"для чтения: %s", path, strerror(errno)

);

(void)\_getch();

exit(1);

}

if (fscanf(file, "%u", &count) != 1)

{

printf(

"\nошибка: файл должен начинаться "

"количеством записей (положительное число)"

);

(void)\_getch();

exit(1);

}

if (count == 0)

{

printf("\nошибка: количество записей не может быть нулевым");

(void)\_getch();

exit(1);

}

orders = (struct order\*)malloc(sizeof(struct order) \* count);

if (orders == NULL) abort();

for (unsigned int i = 0; i < count; i++)

{

if (feof(file))

{

printf(

"\nошибка: файл содержит меньше записей, "

"чем объявлено в его заголовке "

"(%u из %u)", i, count

);

(void)\_getch();

exit(1);

}

if (fscanf(

file,

"%s%s%s%ld%ld",

orders[i].address,

orders[i].kind,

orders[i].date,

&orders[i].weight,

&orders[i].price) != 5)

{

printf(

"\nошибка: запись номер %u повреждена или "

"записана в некорректном формате", i + 1

);

(void)\_getch();

exit(1);

}

insert\_order(orders[i]);

}

while (1) menu();

}

2.1. Спецификация

|  |  |  |  |
| --- | --- | --- | --- |
| Название переменной | Тип | Локальная или глобальная | Описание |
| i,j | int | локальные | Счетчики |
| count | int | глобальная | Число элементов в файле |
| s | char | локальная | Для записи различных символьных строк в функциях |
| y,last\_y | int | локальные | Переменные,необходимые для навигации. |
| found\_match | bool | локальная | Флаговая переменная для завершения работы цикла. |
| str 1 | char | локальная | Для записи имени при построении диаграммы |
| str 2 | char | локальная | Для записи процентов при построении диаграммы. |
| ncolor | int | локальная | Счетчик для изменения цветов при построении диаграммы |
| total | long | локальная | Для нахождении общих затрат при построении диаграммы |

3. Заключение

Программа может быть использована для хранения статистических данных о заказах в сервисах доставки еды. Функции поиска облегчают работу со списком и помогают найти нужную информацию. Это может быть для тех, кто занимается анализом данных о прибыльности сервисов доставки еды. Для эксплуатации приложения необходим компьютер с операционной системой

windows XP (и выше) и оператор.

3.1. Инструкция пользователя

1) Установить файлы kurs.exe и sport.dat в отдельный каталог.

2) Двойным кликом запустить файл kurs.exe, тем самым запустив саму

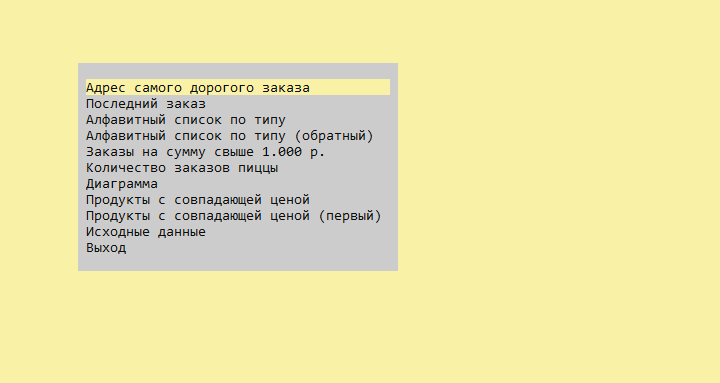
программу.

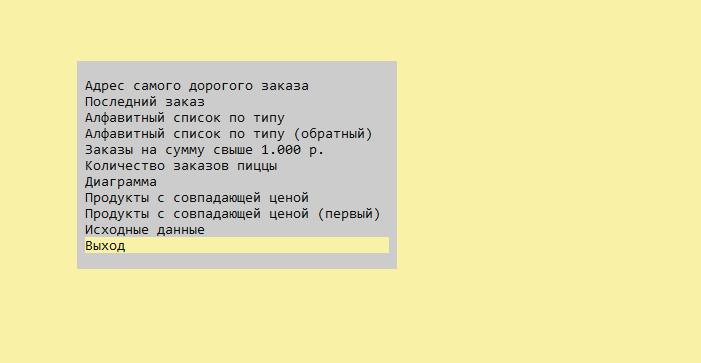
Дальнейшая работа с программой осуществляется через меню. Перемещение по меню осуществляется при помощи кнопок UP (перемещение вверх), DOWN (перемещение вниз), HOME

(перемещение в начало меню) и END (перемещение в конец меню). Далее, будет

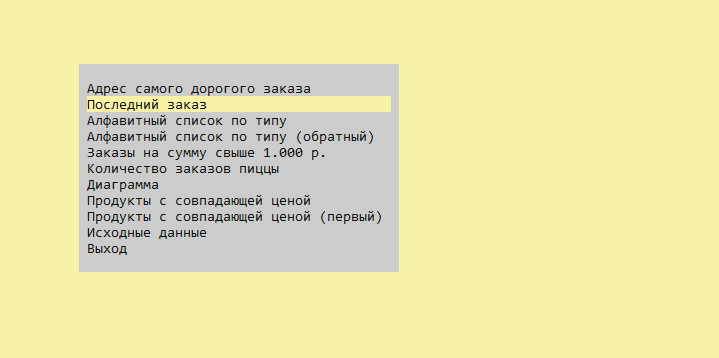
продемонстрирована работа этих клавиш.

Начальный вид меню, сразу после запуска программы:

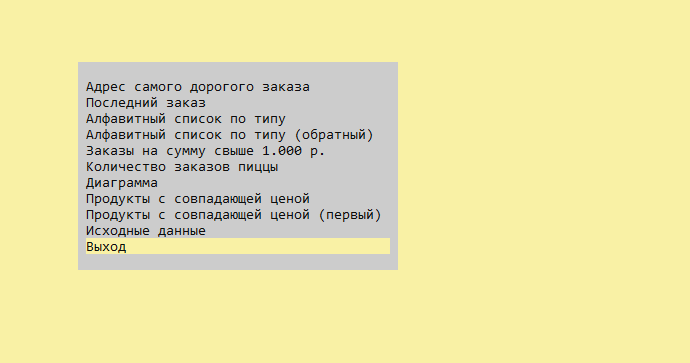
После нажатия кнопки UP:



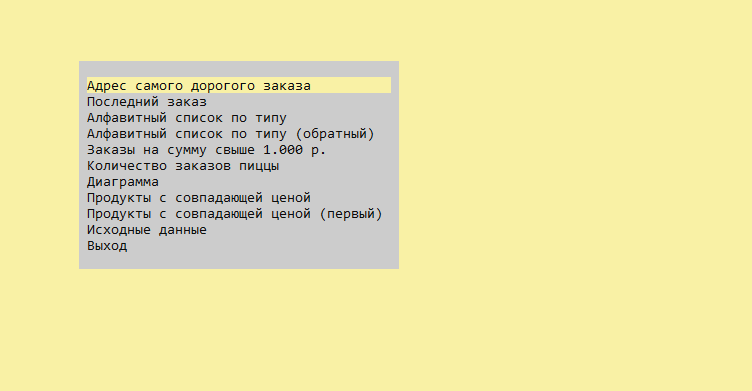
После нажатия кнопки DOWN:

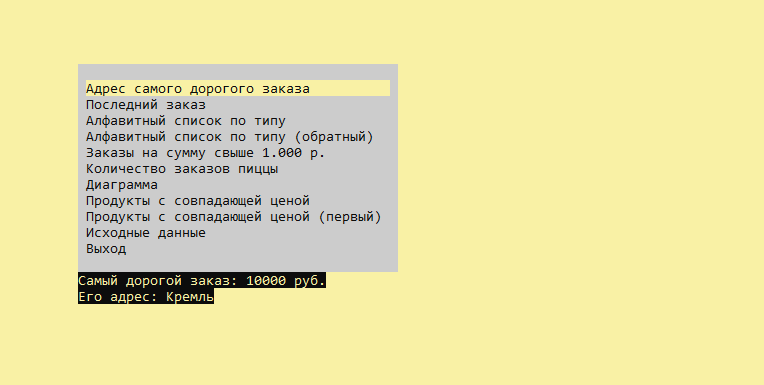


После нажатия кнопки END:



После нажатия кнопки HOME:

Для получения ответа на выбранный вами вопрос, достаточно нажать клавишу ENTER:



Некоторые функции (выводящие список, диаграмму) при нажатии ENTER, перебрасывают пользователя на второй уровень меню – на отдельную страницу.

Чтобы вернуться в меню с вопросами, достаточно нажать любую кнопку на клавиатуре.

4. Список литературы

1. Зоткин, С. П. Программирование на языке высокого уровня: методические указания и варианты заданий для студентов 1-ого курса направления подготовки 09.03.01 Информатика и вычислительная техника / Составители: С. П. Зоткин, - Москва – Московский Государственный строительный университет, Ай Пи Эр Медиа, ЭБС ACB, 2016 – 40 с.
2. Программирование на языке высокого уровня C/C++: конспект лекций / Составители: С. П. Зоткин – 3-е издание – Москва: МИСИ-МГСУ, ЭБС АСВ, 2018 – 140 с.
3. Game Engine Architecture, Third Edition / Составители: Дж. Грегори - 3-е издание - междунар. изд. A K Peters/CRC Press: 2018 – 1240 c.
4. Документация ЯП Си и Си++: https://en.cppreference.com/w/ (дата обращения: 20.06.2024)
5. Документация ЯП Си++: https://cplusplus.com/reference/ (дата обращения: 20.06.2024)
6. Документация Microsoft Windows Forms: https://learn.microsoft.com/en-us/dotnet/desktop/winforms/?view=netdesktop-8.0 (дата обращения: 20.06.2024)
7. Вводный курс языка Си++ на русском: https://www.youtube.com/watch?v=dQw4w9WgXcQ (дата обращения: 30.02.2024)
8. Литвиненко Н. А. Л64 Технология программирования на С++. Начальный курс. — СПб.: БХВ-Петербург, 2005. — 288 с.: ил. ISBN 5-94157-655-2
9. Б. Страуструп Язык программирования С++. Специальное издание. Пер. с англ. — М.: Издательство Бином, 2011 г. — 1136 с.
10. Шилдт Г. Самоучитель C++: Пер. с англ. – 3-е изд. – СПб.: БХВ-Петербург, 2003. – 688 с.
11. Майк МакГрат Программирование на С для начинающих. - 4-е изд. - Эксмо-Пресс, 2016. - 193 с.
12. Александреску, Андрей. Современное проектирование на C++. Серия C++ In-Depth, т. 3.: Пер. С англ. – М.: Издательский дом «Вильямс», 2002. – 336 с.: ил. – Парал. Тит. Англ.
13. Рацеев С. М. Программирование на языке Си. - 1-е изд. - СПб.: Лань, 2022. - 332 с.
14. Стивен Прата. Язык программирования C++ (C++11). Лекции и упражнения, 6-е издание — М.: Вильямс, 2012. — 1248 с.
15. Мейерс С. Эффективное использование С++. 35 новых рекомендаций по улучшению ваших программ и проектов. — М.: ДМК Пресс, 2014. — 294с.