Министерство образования Республики Беларусь

Учреждение образования

БелорусскиЙ государственный университет

информатики и радиоэлектроники

Факультет компьютерных систем и сетей

Кафедра программного обеспечения информационных технологий

|  |
| --- |
|  |
|  |

# ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту по дисциплине:

Основы алгоритмизации и программирования

на тему:

ПРОГРАММНОЕ СРЕДСТВО ДЛЯ ПОСТРОЕНИЯ СХЕМ АЛГОРИТМА ПО ГОСТ 19.701-90

БГУИР КП 6-05-0612-02 101 ПЗ

|  |  |  |  |
| --- | --- | --- | --- |
| Выполнил  Студент гр. 351001 |  | Будников М. А.  М. А. Будников | |
| Проверил |  | Е.Е. Фадеева | |
|  |  | Фадеева Е.Е |
|  |  |  |
|  |  |  |
|  |  |  |

Минск, 2024

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ………………………………………………………………... 3

1. Анализ предметной области………………………………….……….. 4
   1. Анализ аналогов…................……..………………………….. 4
   2. Формирование требований…….....…………………………. 6
2. Проектирование программного средства…………………………….. 7
   1. Структура программы……….………….…………………… 7
   2. Проектирование интерфейса программного средства ……. 7
   3. Проектирование функционала программного средства ….. 9

2.3.1. Главный модуль программы……………………………… 9

2.3.1.1. Графические элементы главного окна……………….. 9

2.3.1.2. Обработчики событий главного окна………………... 10

2.3.2. Подсистема типов данных………………………………... 11

2.3.3. Подсистема отрисовки……………………………………. 12

1. Конструирование программного средства….………………………... 13
   1. Конструирование главного модуля………………………… 13
   2. Конструирование подсистемы данных…………………….. 13
   3. Конструирование подсистемы отрисовки…………………. 14
2. Тестирование программного средства……………………………...... 15
3. Руководство по установке и использованию………………………... 20

ПРИЛОЖЕНИЕ А………………………………….……………………... 25

ПРИЛОЖЕНИЕ Б………………………………….……………………… 35

ВВЕДЕНИЕ

Эффективность и повсеместность использования графического представления алгоритмов создали необходимость в разработке стандартизированных обозначений и правил для построения блок-схем. Стандартом принятом на вооружение в Республике Беларусь является ГОСТ 19.701-90. С существованием стандарта существует и необходимость в инструментах для разработки блок-схем по данному стандарту.

Соответственно, целью моего курсового проекта является создание инструмента, который позволяет строить блок-схемы по ГОСТ-19.701-90. Этим инструментам и является моя программа под кодовым названием ЙЦУКЕН.

В основе работы над ЙЦУКЕН лежат принципы постоянного тестирования добавленного функционала, работа с системой контроля версий Git, использование пройденного в университете материала для реализации технической стороны проекта.

1. **Анализ предметной области**
   1. **Анализ аналогов**

Объектами анализа являются следующие программы: Microsoft Visio 2003, Draw.io.

Таблица 1.1 – Краткая информация об объектах анализа

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование | Разработчик  (сайт) | Назначение | Год выпуска  (версии) |
| Microsoft Visio 2003 | Microsoft (https://www.microsoft.com) | Построение широкого спектра различных блок-схем, диаграмм | 2003 |
| Draw.io | JGraph Ltd | Построение широкого спектра различных блок-схем, диаграмм | 2010 (В 2012 получил название draw.io) |

Таблица 1.2 – Сравнительная таблица Visio 2003 и Draw.io

|  |  |  |  |
| --- | --- | --- | --- |
| т№ | Характеристика | Microsoft Visio 2003 | Draw.io |
| 11 | Операционная система | Microsoft Windows | Любая, необходим лишь доступ к браузеру. Desktop версия для Windows, MacOS, Linux (Debian) |
| 22 | Количество пользователей | Однопользовательское | Однопользовательское |
| 33 | Online или Offline | Offline на компьютере | Online и offline (для offline-режима нужно скачивать Desktop версию) |
| 44 | Время нахождения на рынке | 21 год | 13 лет |

Продолжение таблицы 1.2

|  |  |  |  |
| --- | --- | --- | --- |
| 55 | Известность фирмы-разработчика | Microsoft один из крупнейших технических гигантов, создатели самой популярной ОС для персональных компьютеров: Windows. Компания считается одной из технологических компаний «Большой Пятерки» наряду с Amazon, Meta, Apple и Google | JGraph Ltd – небольшая IT-компания, основанная в Великобритании в 2000 году. Специализируется на программах и технологиях для построения диаграмм на базе браузеров. |
| 66 | Набор выполняемых функций | Выбор категории для построения диаграмм (схемотехника, бизнес, блок-схемы алгоритмов, сети и т.д.)  Поддержка всех символов, необходимых для проектирования блок-схем по ГОСТ-19.701-90 в категории Basic Flowchart.  Управление размером текста, шрифтом, цветом, выравниванием. Наличие различных соединительных элементов (стрелок). | Выбор категории для построения диаграмм (схемотехника, бизнес, блок-схемы алгоритмов, сети и т.д.).  Удобный набор инструментов для создания диаграмм.  Управление размером текста, шрифтом, цветом, выравниванием. Наличие различных соединительных элементов (стрелок). |
| 66.1 | Создание блок-схем | Блок-схемы создаются путем перетаскивания элементов из левого меню на рабочую поверхность. При недостатке места можно создать новые листы. | Блок-схемы создаются путем перетаскивания элементов из левого меню на рабочую поверхность. При недостатке места можно создать новые листы или изменить размер холста. |
| 66.2 | Вебсайт | Для конкретной версии 2003 отсутствует. | https://app.diagrams.net/ |

Продолжение таблицы 1.2

|  |  |  |  |
| --- | --- | --- | --- |
| 66.3 | Мобильное приложение | Отсутствует | Можно попробовать использовать из браузера (неудобно) |
| 66.4 | Платные функции | Отсутствуют |  |
| 66.5 | Дополнительные функции | Интеграция диаграмм, построенных в Visio с другими программами из Microsoft Office | Интеграция диаграмм с облачными сервисами и другими программами. |
| 77 | Функции, которых не хватает данному ПС | Удобное изменение размера холста | Форма фигур немного отличается от ГОСТ-19.701-90 |
| 88 | Функции, которые используют крайне редко | Отсутствуют | Отсутствуют |
| 99 | Стоимость | Условно бесплатно | Бесплатно |

**1.2 Формирование требований**

На основе анализа программных аналогов были составлены следующие требования:

А) Создание программного средства, заточенного под создание блок-схем по ГОСТ 19.701-90.

Б) Наличие следующих функций:

- Настройка размера и шрифта текста, размеров символов.

- Работа со всеми символами из ГОСТ 19.701-90

- Динамическое изменение размеров холста (отсутствует у Visio 2003)

B) Входные данные: файл собственного расширения rog, где хранится информация о всех символах блок-схемы.

Г) Выходные данные: файл собственного расширения rog, файл рисунка png, файл pdf для печати.

1. **Проектирование программного средства**

**2.1 Структура программного средства**

В состав проекта входит 7 модулей:

1. uMain – Главный модуль, внутри которого происходит обработка действий пользователя
2. DataStructures – Модуль, который содержит в себе используемые программой типы данных, методы для взаимодействия с переменными этих типов данных.
3. StackRoutine – Модуль, который добавляет тип данных стек в программу.
4. DrawSymbols – Модуль, ответственный за отрисовку всех графических элементов на холсте.
5. FilesHandling – Модуль для работы с файлами: открытие, сохранение, экспорт.
6. Settings – Модуль, добавляющий окно настроек в программу.

**2.2 Проектирование интерфейса программного средства**

За основу интерфейса взят распространенный шаблон офисных приложений. В самом верху находится панель «Меню» с вкладками: «Файл», «Редактирование», «Режимы» и «Настройки. Действия в этих вкладках продублированы на панели инструментов чуть ниже.

Слева находится панель доступных для перетаскивания символов ГОСТ 19.701-90. По центру экрана располагается холст для размещения символов блок-схемы, а в самом низу окна находится панель статуса, которая отображает текущий режим работы и позицию курсора на экране.

При переходе в Настройки будет отображено окно настроек с возможностью

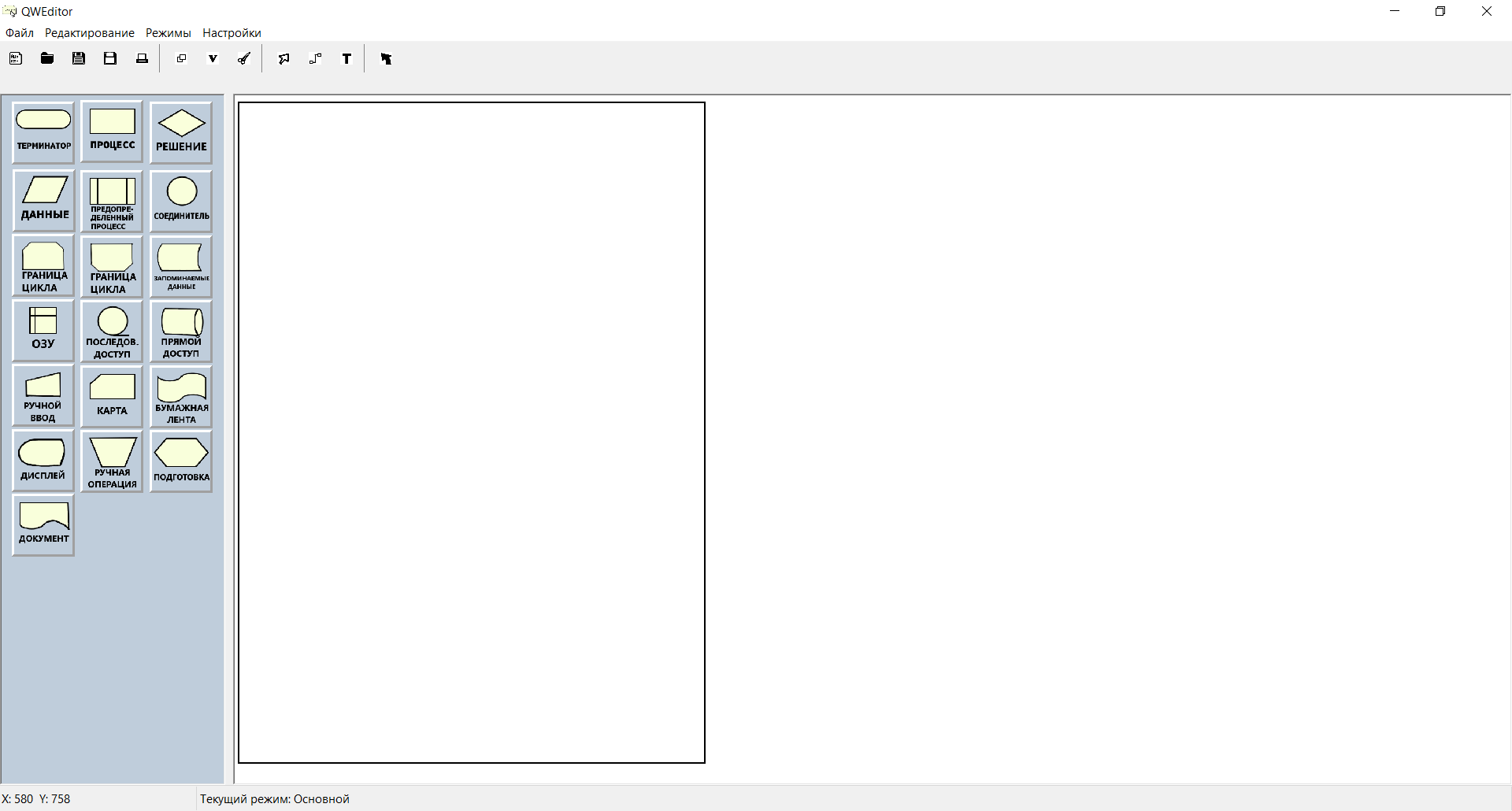
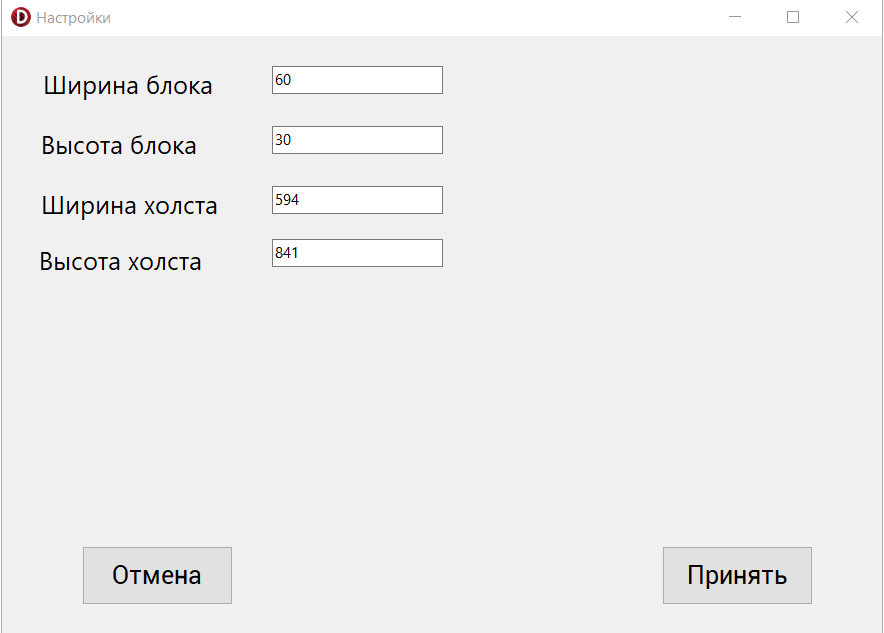
4

Рисунок 2.1 – Окно программы ЙЦУКЕН

При переходе в настройки откроется новое окно, в котором можно изменить

размер символов при перетаскивании и размер холста.

Рисунок 2.2 – окно настроек

**2.3 Проектирование функционала программного средства**

В основе курсового проекта лежат три основные подсистемы:

1. Главный модуль, который в процессе своей работы вызывает необходимые функции, управляет потоком обработки и отображения информации. В программе этоим модулем является uMain.
2. Подсистема типов данных, в которой хранятся описания всех используемых типов данных, методов для работы с ними, методов для реализации допольнительного функционала и функций внутреннего использования. В программе эта подсистема состоит из двух модулей: DataStructures и StackRoutine.
3. Подсистема отрисовки. Данная подсистема имеет различные процедуры для рисования, большая часть которых требует на вход динамический список элементов для отрисовки. Это могут быть символы данных, линий или текстовые метки. В программе задачи этой подсистемы выполняет модуль DrawSymbols.

Например, на основе этих подсистем функционирует модуль для работы с файлами. На вход он получает динамические структуры данных, описанные в подсистеме данных, а на выходе может на основе этих структур с помощью подсистемы отрисовки экспортировать блок-схему в png или pdf.

Примечание: Блок-схемы описываемых алгоритмов находятся в Приложении А.

**2.3.1 Главный модуль программы**

**2.3.1.1 Графические элементы главного окна**

Главный модуль программы по совместительству является и главным окном, которое видит пользователь после запуска программы. Для создания интерфейса используются общепринятые компоненты операционной системы Windows.

Панель «Меню» есть компонент TMainMenu с подключенным компонентом TImageList, в котором хранятся иконки действий, доступных пользователю.

Ниже располагается панель инструментов, которая точно также подключена к компоненту иконок TImageList и которая дублирует доступные в панеле «Меню» действия для более быстрого доступа пользователя к этим действиям.

Слева распологается компонент sbInstruments типаTScrollBox, содержащий в себе панели каждого доступного символа для отрисовки, а эти панели в свою очередь содержат компоненты TImage c изображением соответствующих символов для отрисовки. TScrollBox выбран для того, чтобы при изменении высоты окна можно было пролистнуть панель инструментов вниз и выбрать необходимый символ.

По центру экрана размещается еще один TScrollBox под названием sbMain, который отвечает за содержание в себе компонента pbWorkingArea типа TPaintBox, на котором и происходит отрисовка создаваемой блок-схемы. В случае превышения холстом размеров окна, появляются горизонтальная и вертикальная полосы прокрутки.

В самом низу находится строка состояния, в которой указаны текущий режим работы и позиция курсора мыши на экране.

**2.3.1.2 Обработчики событий главного окна**

Графическая часть, описанная выше, необходима для интуитивно понятного взаимодействия пользователя с программой. С помощью действий нажатия, отпускания, перетягивания и перемещения мыши, нажатия сочетаний клавиш, пользователь дает программе понять, какое действие следует выполнить.

Общие возможные действия пользователя сгруппированы в alMain (TActionList), который хранит элементы типа TAction, разбитые по смысловым категориям. Каждый элемент TAction имеет собственный тег, необходимый для обработчика событий actionsExecuter и индекс связанной с ним иконки из списка иконок ilMain (TImageList).

ActionsExecuter – обработчик всех действий из списка действий alMain. Он проверяет тег отправителя и в зависимости от его значения запускает соответствующие операции.

Действия пользователя, связанные с непосредственным размещением символов на холсте, их выделением и перетаскиванием, перехватываются с помощью обработчиков событий компонента холста pbWorkingArea. Обработчик onMouseDown обрабатывает нажатие мыши в пределах холста, onMouseMove – перемещение, onMouseUp – отжатие мыши, onDragDrop – перетаскивание элементов из панели инструментов на холст. Для определения действия, которое делает пользователь, используется набор флагов и проверка нескольких условий, внутри обработчика onMouseDown (блок-схема в Приложении А).

**2.3.2 Подсистема типов данных**

Все размещенные пользователем элементы хранятся в динамических списках. Я разбил эти элементы на три типа: «Блоки» (TBlock), «Текст» (TText) и линии (TLine). Каждая из этих структур имеет статическую часть типа TBlockInfo, TTextInfo или TLineInfo соответственно для сохранения в файл.

В статической части структур хранится информация, задающая координаты символов, данные, необходимые для отображения, уникальный идентификатор символа. В динамической части хранится указатель на следующий элемент динамического списка и переменная типа TState, которая указывает на состояние текущего символа.

TState является переменной перечислимого типа, где элементы этого типа указывают на состояние символа. Существуют следующие состояния: stNormal (обычное), stSelected (символ выделен, это влияет на его отрисовку и взаимодействие), stLines (во время отрисовки добавляются опорные точки), stText (указывает, что работа происходит в текстовом режиме).

Для изменения и получения данных этих типов разработано целое множество процедур и функций, на вход которым подаются адреса начала соответствующих динамических списков. В Приложении А приведена блок-схема важной функции GetIdByCoord, которая по входным координатам возвращает идентификатор символа, в область которого входят эти координаты.

Для хранения всех предыдущих состояний холста используются структура данных стек. Каждый раз при новом действии, все динамические списки символов сохраняются в новом месте и их указатели добавляются на вершину стека. После отката действий пользователем, эти указатели извлекаются из вершины стека, значения сохраненных списков копируются в динамические списки текущей сессии, а списки из стека очищаются из памяти. Контроль за очисткой памяти динамических списков лежит на вызывающей функции.

**2.3.3 Подсистема отрисовки**

В подсистеме отрисовки содержаться все процедуры, ответственные за рисование графических примитивов на экране. Для каждого вида символов существует отдельная процедура, которая строит его по заданным левому верхнему и правому нижнему углам. Основной процедурой этой подсистемы является процедура DrawAll, которая принимает на вход холст типа TCanvas, на котором необходимо отрисовать символы и динамические списки «блоков», «текста» и «линий». Блок-схема данной процедуры находится в Приложении А.

Процедура DrawSelection рисует вокруг символа на определенном расстоянии пунтктирную обводку зеленого цвета для лучшей видимости выделенных символов.

Линии рисуются с помощью процедуры DrawLine, которая по заданному массиву координат проводит прямые.

Текст выводится с помощью функции операционной системы Windows DrawText. Эта функция позволяет центрировать текст по горизонтали и определять границы его отображения.

Так как всем процедурам подсистемы отрисовки передается холст для рисования, это позволяет отрисовывать блок-схему не только на главном экране, видимом пользователю, но и на временно создаваемые холсты у типа TBitMap для дальнейшего сохранения в файлы pdf и png.

1. **Конструирование программного средства**
   1. **Конструирование главного модуля**

Для реализации описанного в разделе «Проектирование программного средства» функционала возникла необходимость в создании целого набора приватных переменных главного модуля, которые регулируют основные характеристики, при которых будет происходить то или иное действия. Инициализация всех этих переменных происходит после создания окна в обработчике событий onFormCreate главной формы.

Размер холста по умолчанию устанавливается в стандарт А4. Инициализируются все динамические типы данных, устанавлиются размеры символов по умолчанию, режим работы, теги действий типа TAction и теги перетаскиваемых картинок типа TImage из панели инструментов.

Своих внутренних подпрограмм главный модуль не имеет, используются обработчики событий компонентов формы и импортируемые подпрограммы из других модулей.

Код всех модулей программного средства находится в Приложении Б.

* 1. **Конструирование подсистемы данных.**

В подсистеме данных размещаются все базовые подпрограммы для создания, изменения, удаления и получения данных из разработанных динамических структур данных стека и однонаправленного списка.

Однако в подсистеме данных находятся не только и не столько базовые подпрограммы. В ходе разработки программного средства постоянно возникала необходимость в написании новых подпрограмм для взаимодействия с существующими типами данных. Так появились функции GetLabelIdByCoord и GetBlockIdByCoord, которые возвращают идентификатор текстовой метки или блока, внутри которого находится входная координата. То есть, появились ситуации, в которых стало необходимо отделять типы входящих символов друг от друга.

Во время разработки дополнительного функционала выделения символов в выделенной мышью области появилась необходимость в процедуре SelectSymbolsInArea, которая проходится по всем символам на холсте и проверяет, входят ли они в выделенную область. Если входят, то переменная их состояния state меняется на stSelected и при следующем отображении этих символов на экран они будут выделены.

Для реализации перемещения выделенных символов с зажатой левой клавишей мыши была создана процедура OffsetSelectedSymbols, которая изменяет координаты левого верхнего и правого нижнего углов каждого выделенного символа на смещение равное входным данным.

В программе существуют события, которые влияют на все созданные символы. Для одновременного изменения состояния каждого символа была создана процедура SetSymbolsState. В основном она используется для выделения всех символов событием actSelectAll (горячая клавиша Ctrl+A), выхода из выделения и перехода в режим линий.

Для реализации буфера обмена и действия отката изменений была создана процедура CopySymbolsFromTo, которая принимает на вход шесть динамических типов: источники и конечные адресса, куда необходимо скопировать данные.

При построении линии изначально известны лишь ее координата начала и конца. Чтобы из этих данных построить массив координат точек линии, была создана процедура ConstructLine.

Код почти всей подсистемы данных находится в модуле DataStructures (кроме реализации стека, она находится в модуле StackRoutine) в Приложении Б.

**3.3** **Конструирование подсистемы отрисовки.**

Так как в ГОСТ 19.701-90 существует внушительное количество различных символов, то для каждого из них пришлось написать отдельную процедуру отрисовки с помощью встроенных в Delphi VCL процедур отрисовки геометрических примитивов.

Для большинства символов хватало встроенных процедур Polygon (рисование многоугольника по заданным точкам) и Rectangle (рисование прямоугольника по левому верхнему и правому нижнему углу). Некоторые символы требовали использования Ellipse (рисование эллипса, вписанного в прямоугольник, координаты углов которого передаются входными данными). Однако следующие символы: «Документ», «Запоминаемые данные», «Прямой доступ», «Бумажная лента» и «Дисплей» потребовали более сложных процедур отрисовки с использованием встроенной в Delphi VCL процедуры рисования дуги Arc.

Для установления параметров кисти рисования на холсте создана процедура SetCanvaAttributes, которая заданному холсту меняет параметры кисти в зависимости от входной переменной состояния типа TState.

Код всей подсистемы отрисовки находится в модуле DrawSymbols в Приложении Б.

1. **Тестирование программного средства**

Во время разработки программы проводилось постоянное тестирование нововведенного функционала на его работоспособность. Следующие несколько тестов представляют собой тестирование уже готового программного средства на соответствие техническим требованиям и правильности работы созданных функций.

* 1. **Текст программы**

**Основное тело программы (Main.dpr):**

program Main;

{$APPTYPE CONSOLE}

{$R \*.res}

uses

System.SysUtils,

windows,

System.Types,

FilesHandling,

MonodirectionalList,

Include,

GiftGenerator;

var

presents: PSurprize;

procedure Create(var list: PGroup); forward;

// Application logic

procedure Start(var groupToInit: PGroup); forward;

procedure Finish(groups: PGroup); forward;

function ReadSessionManager(groups: PGroup): integer; forward;

function CloseSessionManager(groups: PGroup): integer; forward;

function KeyController(groups: PGroup): integer; forward;

procedure HandleEvent(event: integer); forward;

function AddManager(groups: PGroup): integer; forward;

function EditManager(groups: PGroup): integer; forward;

function DeleteManager(groups: PGroup): integer; forward;

function PrintManager(groups: PGroup): integer; forward;

function SearchManager(groups: PGroup): integer; forward;

function SortManager(groups: PGroup): integer; forward;

function PresentManager(groups: PGroup): integer; forward;

procedure AddSweetWrapper(groups: PGroup); forward;

function Search(groups: PGroup; code: integer; bag: PSweet): integer; forward;

procedure Print(list: PGroup); forward;

procedure PrintAll(list: PGroup); forward;

// Interface

procedure ClearScreen(keepFirstLine: boolean = false); forward;

procedure ClearLine(y: integer); forward;

procedure PutCursorToPos(x, y: integer); forward;

procedure SetColor(bgColor, fgColor: integer); forward;

procedure ShowMessage(y: integer; errorText: string;

messageType: integer = 0); forward;

procedure ShowMainMenu(); forward;

procedure ShowMenuBar(const actions: array of string); forward;

procedure ShowTableHeader(); forward;

procedure ShowSweet(sweet: PSweet); forward;

// FUNCTIONS AND PROCEDURES

function AddManager;

var

response: string;

code: integer;

begin

Result := OK\_CODE;

ShowMenuBar(['Добавить группу', 'Добавить сладость', 'Назад']);

readln(response);

ClearLine(INPUT\_LINE);

if TryStrToInt(response, code) then

begin

case code of

1:

begin

ShowMenuBar([' ДОБАВЛЕНИЕ НОВОЙ ГРУППЫ ']);

writeln;

write('Наименование новой группы: ');

SetColor(clBlack, clYellow);

readln(response);

if not((response = 'q') or (response = '')) then

begin

AddGroup(groups, response);

ShowMessage(3, 'Группа успешно была добавлена!', 2);

end

else

ShowMessage(3,

'Заsпись не была добавлена. Нажмите ENTER чтобы выйти', 1)

end;

2:

begin

ShowMenuBar

([' ДОБАВЛЕНИЕ НОВОЙ СЛАДОСТИ (Чтобы отменить добавление, введите q в любом поле ввода) ']

);

AddSweetWrapper(groups);

end;

3:

Result := OK\_CODE;

else

Result := INVALID\_VALUE\_CODE;

end;

end

else

Result := INVALID\_INTEGER\_CODE;

ClearScreen(true);

ShowMainMenu();

end;

procedure AddSweetWrapper;

var

sweet: PSweet;

strVar: string;

intVar: integer;

currentLine: integer;

stayFlag: boolean;

correctInputFlag: boolean;

groupID: integer;

hOut: THandle;

begin

hOut := GetStdHandle(STD\_OUTPUT\_HANDLE);

currentLine := 1;

stayFlag := true;

strVar := '';

New(sweet);

correctInputFlag := false;

while not(correctInputFlag) do

begin

write(#13#10'ID или Название группы для добавления сладости: ');

SetColor(clBlack, clYellow);

readln(strVar);

SetColor(clBlack, clWhite);

ClearLine(2);

if ((strVar = 'q') or (strVar = '')) then

begin

stayFlag := false;

correctInputFlag := true;

end

else if TryStrToInt(strVar, groupID) then

begin

if isGroupExists(groups, groupID) then

correctInputFlag := true

else

ShowMessage(currentLine, 'Данной группы не существует!');

end

else

begin

groupID := GetGroupIDByName(groups, strVar);

if groupID = -1 then

ShowMessage(currentLine, 'Данной группы не существует!')

else

correctInputFlag := true;

end;

end;

if stayFlag then

begin

inc(currentLine);

write('Название сладости: ');

readln(strVar);

end;

if ((strVar = 'q') or (strVar = '')) then

stayFlag := false;

if stayFlag then

begin

inc(currentLine);

correctInputFlag := false;

sweet.info.name := strVar;

while not(correctInputFlag) do

begin

write('Цена: ');

readln(strVar);

if TryStrToInt(strVar, intVar) then

correctInputFlag := true

else

begin

if strVar = 'q' then

begin

stayFlag := false;

correctInputFlag := true;

end

else

ShowMessage(currentLine, 'Введено некорректное значение!');

end;

end;

end;

if stayFlag then

begin

inc(currentLine);

correctInputFlag := false;

sweet.info.price := intVar;

while not(correctInputFlag) do

begin

write('Вес: ');

readln(strVar);

if TryStrToInt(strVar, intVar) then

correctInputFlag := true

else

begin

if strVar = 'q' then

begin

stayFlag := false;

correctInputFlag := true;

end

else

ShowMessage(currentLine, 'Введено некорректное значение!');

end;

end;

end;

if stayFlag then

begin

inc(currentLine);

correctInputFlag := false;

sweet.info.weight := intVar;

while not(correctInputFlag) do

begin

write('Сахар: ');

readln(strVar);

if TryStrToInt(strVar, intVar) then

correctInputFlag := true

else

begin

if strVar = 'q' then

begin

stayFlag := false;

correctInputFlag := true;

end

else

ShowMessage(currentLine, 'Введено некорректное значение!');

end;

end;

end;

inc(currentLine);

if not(stayFlag) then

ShowMessage(currentLine,

'Запись не была добавлена. Нажмите ENTER чтобы выйти', 1)

else

begin

sweet.info.sugar := intVar;

AddSweet(groups, groupID, sweet);

ShowMessage(currentLine,

'Запись была успешно добавлена! Нажмите ENTER чтобы выйти', 2);

end;

end;

procedure ClearLine;

begin

PutCursorToPos(0, y);

writeln(StringOfChar(' ', MAX\_WIDTH));

PutCursorToPos(0, y);

end;

procedure ClearScreen;

var

ConsoleSize, NumWritten: LongWord;

Origin: COORD;

ScreenBufferInfo: CONSOLE\_SCREEN\_BUFFER\_INFO;

hOut: THandle;

begin

hOut := GetStdHandle(STD\_OUTPUT\_HANDLE);

GetConsoleScreenBufferInfo(hOut, ScreenBufferInfo);

Origin.x := 0;

if keepFirstLine then

begin

ConsoleSize := ScreenBufferInfo.dwSize.x \* (ScreenBufferInfo.dwSize.y - 1);

Origin.y := 1;

end

else

begin

ConsoleSize := ScreenBufferInfo.dwSize.x \* ScreenBufferInfo.dwSize.y;

Origin.y := 0;

end;

FillConsoleOutputCharacter(hOut, ' ', ConsoleSize, Origin, NumWritten);

FillConsoleOutputAttribute(hOut, ScreenBufferInfo.wAttributes, ConsoleSize,

Origin, NumWritten);

SetConsoleCursorPosition(hOut, Origin);

end;

function CloseSessionManager;

var

response: string;

response2: string;

actionCode: integer;

begin

Result := EXIT\_CODE;

ShowMenuBar([' ВЫХОД С СОХРАНЕНИЕМ ']);

write('Название сессии для сохранения <Enter или q чтобы вернуться в главное меню>: ');

readln(response);

ClearLine(1);

if (response = '') or (response = 'q') then

begin

Result := OK\_CODE;

ShowMainMenu();

end

else

begin

if Length(response) > 30 then

begin

Result := LENGTH\_EXCEED\_CODE;

ClearScreen();

ShowMainMenu();

end

else

begin

if isSessionExists(response) then

begin

ShowMessage(1, 'Сессия с таким названием уже существует!', 1);

write('Введите "1", чтобы всё равно сохранить и выйти, либо Enter, чтобы вернуться в главное меню: ');

readln(response2);

ClearLine(1);

if (TryStrToInt(response2, actionCode) = true) and (actionCode = 1) then

begin

SaveSession(groups, response);

ShowMessage(1, 'Сессия "' + response + '" успешно сохранена!', 2);

end

else

begin

Result := OK\_CODE;

ShowMainMenu();

end;

end

else

begin

SaveSession(groups, response);

ShowMessage(1, 'Сессия "' + response + '" успешно сохранена!', 2);

end;

end;

end;

end;

procedure Create;

var

i, k: integer;

temp: PGroup;

sweetList: PSweet;

begin

New(list);

New(list^.sweets);

list^.sweets.info.id := 0;

temp := list;

temp.info.id := 0;

for i := 1 to Length(groupNames) do

Begin

New(list^.next);

inc(temp.info.id);

list := list.next;

list.info.id := temp.info.id;

list.info.name := groupNames[i - 1];

End;

list^.next := nil;

list := temp^.next;

k := 0; // Index of current group

while list <> nil do

begin

New(list.sweets);

sweetList := list^.sweets;

sweetList.info.id := 0;

for i := 1 to Length(sweetNames[k]) do

begin

New(sweetList^.next);

inc(temp^.sweets.info.id);

sweetList := sweetList^.next;

sweetList.info.id := temp^.sweets.info.id;

sweetList.info.name := sweetNames[k][i - 1];

sweetList.info.price := sweetPrices[k][i - 1];

sweetList.info.weight := sweetWeights[k][i - 1];

sweetList.info.sugar := sweetSugars[k][i - 1];

end;

list := list^.next;

inc(k);

end;

list := temp;

end;

function DeleteManager;

var

response: string;

code: integer;

correctInputFlag: boolean;

stayFlag: boolean;

id: integer;

bag: PSweet;

begin

Result := OK\_CODE;

correctInputFlag := false;

stayFlag := true;

New(bag);

ShowMenuBar(['Удалить группу', 'Удалить сладость', 'Назад']);

readln(response);

ClearLine(INPUT\_LINE);

if TryStrToInt(response, code) then

begin

case code of

1:

begin

while not(correctInputFlag) do

begin

PutCursorToPos(0, 2);

write('ID или Название удаляемой группы (q чтобы выйти): ');

SetColor(clBlack, clYellow);

readln(response);

SetColor(clBlack, clWhite);

ClearLine(2);

if ((response = 'q') or (response = '')) then

begin

correctInputFlag := true;

stayFlag := false;

end

else if TryStrToInt(response, id) then

begin

if isGroupExists(groups, id) then

correctInputFlag := true

else

ShowMessage(2, 'Данной группы не существует!');

end

else

begin

id := GetGroupIDByName(groups, response);

if id = -1 then

ShowMessage(2, 'Данной группы не существует!')

else

correctInputFlag := true;

end;

end;

if stayFlag then

begin

if DeleteGroup(groups, id) then

ShowMessage(2, 'Группа была успешно удалена!', 2)

else

ShowMessage(2, 'Группы к удалению не существует!', 1);

end;

end;

2:

begin

while not(correctInputFlag) do

begin

PutCursorToPos(0, 2);

write('ID или Название удаляемой сладости (q чтобы выйти): ');

SetColor(clBlack, clYellow);

readln(response);

SetColor(clBlack, clWhite);

ClearLine(2);

if ((response = 'q') or (response = '')) then

begin

correctInputFlag := true;

stayFlag := false;

end

else if TryStrToInt(response, id) then

begin

if isSweetExists(groups, id) then

correctInputFlag := true

else

ShowMessage(2, 'Данной сладости не существует!');

end

else

begin

id := GetSweetIDByName(groups, response);

if id = -1 then

ShowMessage(2, 'Данной сладости не существует!')

else

begin

bag.info.name := response;

if Search(groups, CMP\_NAME, bag) > 1 then

begin

write('Обнаружено несколько записей, укажите индекс для удаления: ');

readln(response);

if not(TryStrToInt(response, id)) then

begin

stayFlag := false;

Result := INVALID\_INTEGER\_CODE;

end;

end;

correctInputFlag := true;

end;

end;

end;

if stayFlag then

begin

if DeleteSweet(groups, id) then

ShowMessage(2, 'Сладость была успешно удалена!', 2)

else

ShowMessage(2, 'Сладости к удалению не существует!', 1);

end;

end;

3:

Result := OK\_CODE;

else

Result := INVALID\_VALUE\_CODE;

end;

end

else

Result := INVALID\_INTEGER\_CODE;

Dispose(bag);

ClearScreen(true);

ShowMainMenu();

end;

function EditManager;

var

sweetToChange, bag: PSweet;

strVar: string;

intVar: integer;

currentLine: integer;

stayFlag: boolean;

correctInputFlag: boolean;

id: integer;

operation: integer;

begin

currentLine := 2;

Result := OK\_CODE;

stayFlag := true;

correctInputFlag := false;

strVar := '';

New(bag);

ShowMenuBar

([' РЕДАКТИРОВАНИЕ ПОЗИЦИЙ (нажмите q или введите пустую строку, чтобы выйтин на любом этапе) ']

);

while not(correctInputFlag) do

begin

PutCursorToPos(0, currentLine);

write('ID или Название сладости для редактирования: ');

SetColor(clBlack, clYellow);

readln(strVar);

SetColor(clBlack, clWhite);

ClearLine(currentLine);

if ((strVar = 'q') or (strVar = '')) then

begin

stayFlag := false;

correctInputFlag := true;

ShowMessage(currentLine, 'Изменение позиции было отменено!', 1);

end

else if TryStrToInt(strVar, id) then

begin

sweetToChange := ExtractSweetByID(groups, id);

if sweetToChange.info.id <> -1 then

correctInputFlag := true

else

begin

stayFlag := false;

correctInputFlag := false;

ShowMessage(currentLine, 'Данной сладости не существует!');

end;

end

else

begin

bag.info.name := strVar;

if Search(groups, CMP\_NAME, bag) > 1 then

begin

write('Обнаружено несколько записей, укажите индекс для редактирования: ');

readln(strVar);

if not(TryStrToInt(strVar, id)) then

begin

correctInputFlag := true;

stayFlag := false;

Result := INVALID\_INTEGER\_CODE;

id := -1

end;

end

else

id := GetSweetIDByName(groups, strVar);

if Result = OK\_CODE then

begin

sweetToChange := ExtractSweetByID(groups, id);

if sweetToChange.info.id <> -1 then

correctInputFlag := true

else

begin

stayFlag := false;

// correctInputFlag := true;

ShowMessage(currentLine, 'Данной сладости не существует!');

end;

end;

ClearScreen(true);

end;

end;

if stayFlag then

begin

ShowMenuBar(['Редактировать название', 'Цену', 'Вес', 'Сахар',

'Вернуться в главное меню']);

writeln('Позиция к изменению:');

ShowSweet(sweetToChange);

currentLine := 5;

correctInputFlag := false;

while not(correctInputFlag) do

begin

readln(strVar);

if TryStrToInt(strVar, operation) then

begin

case operation of

1:

begin

currentLine := 6;

write('Введите новое значение: ');

readln(strVar);

if ((strVar = 'q') or (strVar = '')) then

begin

stayFlag := false;

correctInputFlag := true;

ShowMessage(currentLine, 'Изменение позиции отменено!', 1);

end;

if stayFlag then

begin

correctInputFlag := true;

sweetToChange.info.name := strVar;

writeln('Измененная позиция:');

ShowSweet(sweetToChange);

currentLine := currentLine + 5;

ShowMessage(currentLine,

' Позиция была успешно изменена! ', 2);

end;

end;

2 .. 4:

begin

currentLine := 6;

while not(correctInputFlag) do

begin

write('Введите новое значение: ');

readln(strVar);

ClearLine(currentLine);

if ((strVar = 'q') or (strVar = '')) then

begin

stayFlag := false;

correctInputFlag := true;

ShowMessage(currentLine, 'Изменение позиции отменено!', 1);

end

else if TryStrToInt(strVar, intVar) then

begin

correctInputFlag := true;

case operation of

2:

sweetToChange.info.price := intVar;

3:

sweetToChange.info.weight := intVar;

4:

sweetToChange.info.sugar := intVar;

end;

writeln('Измененная позиция:');

ShowSweet(sweetToChange);

currentLine := currentLine + 5;

ShowMessage(currentLine,

' Позиция была успешно изменена! ', 2);

end

else

ShowMessage(currentLine,

' Введено нечисловое значение! ');

end;

end;

5:

correctInputFlag := true;

else

ShowMessage(currentLine, 'Введен несуществующий пункт меню!');

end;

end

else

ShowMessage(currentLine, 'Введено нечисловое значение!');

end;

end;

ClearScreen();

ShowMainMenu();

end;

procedure Finish;

begin

Dispose(groups); // Zone to correct

DisposePresents(presents);

end;

procedure HandleEvent;

begin

PutCursorToPos(0, 1);

case event of

INVALID\_VALUE\_CODE:

ShowMessage(1, 'Вы ввели несуществующий пункт меню!');

INVALID\_INTEGER\_CODE:

ShowMessage(1, 'Вы ввели нечисловое значение!');

INVALID\_PERCENT\_CODE:

ShowMessage(1,

'Значение процентного содержания должно быть от 0 до 100!');

FEATURE\_NOT\_EXISTS\_CODE:

ShowMessage(1, 'Данного пункта еще не существует. ', 1);

UNKNOWN\_ERROR\_CODE:

ShowMessage(1, 'Неизвестная ошибка ', 1);

SESSION\_NOT\_FOUND\_CODE:

ShowMessage(1, 'Сессии с таким названием не существует! ', 1);

LENGTH\_EXCEED\_CODE:

ShowMessage(1, 'Превышена длина допустимого ввода! ');

INVALID\_INPUT\_CODE:

ShowMessage(1, 'Введены неверные данные! ');

EXIT\_CODE:

begin

ClearScreen();

PutCursorToPos(MAX\_WIDTH div 3, MAX\_HEIGHT div 3);

SetColor(clBlack, clYellow);

writeln('СПАСИБО ЗА ИСПОЛЬЗОВАНИЕ БЕЛОГО ЭНДОРФИНА!');

PutCursorToPos(MAX\_WIDTH div 3, MAX\_HEIGHT div 3 + 1);

writeln(' (НАЖМИТЕ ENTER ЧТОБЫ ВЫЙТИ) ');

SetColor(clBlack, clWhite);

end;

end;

end;

function KeyController;

var

response: string;

code: integer;

begin

Result := OK\_CODE;

PutCursorToPos(0, 1);

readln(response);

ClearLine(INPUT\_LINE);

if TryStrToInt(response, code) then

begin

case code of

1:

Result := ReadSessionManager(groups);

2:

Result := PrintManager(groups);

3:

Result := SortManager(groups);

4:

Result := SearchManager(groups);

5:

Result := AddManager(groups);

6:

Result := DeleteManager(groups);

7:

Result := EditManager(groups);

8:

Result := PresentManager(groups);

9:

Result := EXIT\_CODE;

10: // May be 0

Result := CloseSessionManager(groups);

else

Result := INVALID\_VALUE\_CODE;

end;

end

else

Result := INVALID\_INTEGER\_CODE;

end;

function PresentManager;

var

response, strVar: string;

id, countTypes, PriceMax, WeightMax: integer;

present: PSurprize;

begin

Result := OK\_CODE;

write('Введите количество типов сладостей в подарках: ');

readln(response);

if TryStrToInt(response, countTypes) and (countTypes <= GetGroupsCount(groups))

then

begin

write('Введите макс. цену подарка: ');

readln(response);

if TryStrToInt(response, PriceMax) and (PriceMax <= 10000) then

begin

write('Введите макс. вес подарка: ');

readln(response);

if TryStrToInt(response, WeightMax) and (WeightMax <= 10000) then

begin

DisposePresents(presents);

presents := MainHandler(groups, PriceMax, WeightMax, countTypes);

ClearScreen(true);

if GlobalCount > 0 then

begin

SortPresents(presents);

SavePresents(presents, 'presents.txt');

ShowMessage(1, 'Успешно создано ' + IntToStr(GlobalCount) +

' подарков!', 2);

end

else

ShowMessage(1, 'Подарков для данных условий не существует!', 1);

end

else

Result := INVALID\_INPUT\_CODE;

end

else

Result := INVALID\_INPUT\_CODE;

end

else

Result := INVALID\_INPUT\_CODE;

ClearScreen(true);

end;

procedure Print;

begin

PutCursorToPos(0, 2);

list := list^.next;

writeln(' --------------------------------------------- ');

writeln('| id | Group |');

writeln(' --------------------------------------------- ');

while list <> nil do

begin

writeln('|', list.info.id:7, ' |', list^.info.name:30, ' |');

list := list^.next;

writeln(' --------------------------------------------- ');

end;

PutCursorToPos(0, 0);

PutCursorToPos(0, 1);

readln;

ClearScreen(true);

end;

procedure PrintAll;

var

sweetList: PSweet;

begin

PutCursorToPos(0, 2);

ShowTableHeader();

while list.next <> nil do

begin

list := list.next;

sweetList := list.sweets;

writeln('| ', list^.info.name:30,

' |');

writeln('|----------------------------------------------------------------------------|');

while sweetList^.next <> nil do

begin

sweetList := sweetList^.next;

writeln('|', sweetList.info.id:7, ' |', sweetList.info.name:30, ' |',

sweetList.info.price:8, ' |', sweetList.info.weight:7, ' |',

sweetList.info.sugar:7, ' |');

writeln('|----------------------------------------------------------------------------|');

end;

end;

PutCursorToPos(0, 0);

PutCursorToPos(0, 1);

readln;

ClearScreen(true);

end;

function PrintManager;

var

response: string;

code: integer;

begin

Result := OK\_CODE;

ShowMenuBar(['Отобразить группы', 'Отобразить все сладости',

'Отобразить подарки', 'Отобразить все сладости в подарках', 'Назад']);

readln(response);

ClearLine(INPUT\_LINE);

if TryStrToInt(response, code) then

begin

case code of

1:

Print(groups);

2:

PrintAll(groups);

3:

Result := FEATURE\_NOT\_EXISTS\_CODE;

4:

Result := FEATURE\_NOT\_EXISTS\_CODE;

5:

Result := OK\_CODE;

else

Result := INVALID\_VALUE\_CODE;

end;

end

else

Result := INVALID\_INTEGER\_CODE;

ShowMainMenu();

end;

procedure PutCursorToPos;

var

Origin: COORD;

begin

Origin.x := x;

Origin.y := y;

SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE), Origin);

end;

function ReadSessionManager;

var

folder: string;

folderNumber: integer;

sessions: TStringDynArray;

begin

if isSessionOpen then

ShowMessage(1, 'У вас уже открыта текущая сессия!')

else

begin

ShowMenuBar

(['ВВЕДИТЕ НАЗВАНИЕ СЕССИИ (q или Enter чтобы вернуться в главное меню)']

);

ShowExistingSessions();

sessions := GetExistingSessions();

write('Сессия: ');

readln(folder);

ClearScreen(true);

if (folder = '') or (folder = 'q') then

Result := OK\_CODE

else

begin

if TryStrToInt(folder, folderNumber) = true then

begin

if (folderNumber > 0) and (folderNumber <= Length(sessions)) then

folder := sessions[folderNumber - 1]

else

Result := INVALID\_VALUE\_CODE;

end;

if OpenSession(groups, folder) = OK\_CODE then

begin

ShowMessage(1,

'Сессия открыта, нажмите Enter чтобы вернуться в главное меню', 2);

Result := OK\_CODE;

end

else

Result := SESSION\_NOT\_FOUND\_CODE;

end;

ShowMainMenu();

end;

end;

function Search;

var

sweet1: PSweet;

intVar: integer;

foundFlag: boolean;

begin

Result := 0;

foundFlag := false;

writeln;

ShowTableHeader();

while groups^.next <> nil do

begin

groups := groups^.next;

sweet1 := groups^.sweets;

while sweet1.next <> nil do

begin

sweet1 := sweet1.next;

if CompareSweetFields(sweet1, bag, code) = 0 then

begin

foundFlag := true;

inc(Result);

writeln('|', sweet1.info.id:7, ' |', sweet1.info.name:30, ' |',

sweet1.info.price:8, ' |', sweet1.info.weight:7, ' |',

sweet1.info.sugar:7, ' |');

writeln('|----------------------------------------------------------------------------|');

end;

end;

end;

if foundFlag = false then

begin

writeln('| ПОЗИЦИЙ С ЗАДАННЫМ ПОЛЕМ НЕ НАЙДЕНО! |');

writeln('|----------------------------------------------------------------------------|');

end;

PutCursorToPos(0, 0);

PutCursorToPos(0, 2);

end;

function SearchManager;

var

bag: PSweet;

response: string;

code, intVar: integer;

correctInputFlag: boolean;

begin

Result := OK\_CODE;

correctInputFlag := true;

New(bag);

ShowMenuBar(['Код сладости', 'Наименование', 'Цена',

'Верхняя граница процентного содержания сахара', 'Назад']);

readln(response);

ClearLine(INPUT\_LINE);

if TryStrToInt(response, code) then

begin

case code of

1:

begin

ShowMenuBar([' ПОИСК ПО ID ']);

write('ID для поиска: ');

readln(response);

if TryStrToInt(response, intVar) then

bag.info.id := intVar

else

begin

correctInputFlag := false;

Result := INVALID\_INTEGER\_CODE;

end;

end;

2:

begin

ShowMenuBar([' ПОИСК ПО НАИМЕНОВАНИЮ ']);

write('Название для поиска: ');

readln(response);

bag.info.name := response;

end;

3:

begin

ShowMenuBar([' ПОИСК ПО ВЕРХНЕЙ ГРАНИЦЕ ЦЕНЫ ']);

write('Значение для поиска: ');

readln(response);

if TryStrToInt(response, intVar) then

bag.info.price := intVar

else

begin

correctInputFlag := false;

Result := INVALID\_INTEGER\_CODE;

end;

end;

4:

begin

ShowMenuBar([' ПОИСК ПО ПРОЦЕНТНОМУ СОДЕРЖАНИЮ САХАРА ']);

write('Значение для поиска: ');

readln(response);

if TryStrToInt(response, intVar) then

begin

if ((intVar < 0) or (intVar > 100)) then

begin

correctInputFlag := false;

Result := INVALID\_PERCENT\_CODE;

end

else

bag.info.sugar := intVar;

end

else

begin

Result := INVALID\_PERCENT\_CODE;

correctInputFlag := false;

end;

end;

5:

begin

Result := OK\_CODE;

correctInputFlag := false;

end

else

Result := INVALID\_VALUE\_CODE;

correctInputFlag := false;

end;

end

else

begin

Result := INVALID\_INTEGER\_CODE;

correctInputFlag := false;

end;

if correctInputFlag then

begin

Result := Search(groups, code, bag);

readln;

if Result >= 0 then

Result := OK\_CODE

else

Result := UNKNOWN\_ERROR\_CODE;

end;

Dispose(bag);

ClearScreen(true);

ShowMainMenu();

end;

procedure SetColor;

var

hOut: THandle;

begin

hOut := GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hOut, bgColor \* 16 + fgColor)

end;

procedure ShowMessage;

begin

ClearLine(y);

case messageType of

0:

SetColor(clRed, clWhite);

1:

SetColor(clBrown, clWhite);

2:

SetColor(clGreen, clWhite);

end;

write(errorText);

SetColor(clBlack, clWhite);

readln;

ClearLine(y);

end;

procedure ShowMenuBar;

var

i: integer;

hOut: THandle;

begin

hOut := GetStdHandle(STD\_OUTPUT\_HANDLE);

ClearLine(0);

// Low(actions) = 0

for i := 1 to Length(actions) do

begin

SetConsoleTextAttribute(hOut, $F1);

write(' ', i, ' ');

SetConsoleTextAttribute(hOut, $1F);

write(' ', actions[i - 1], ' ');

end;

writeln;

SetConsoleTextAttribute(hOut, 7);

end;

procedure ShowMainMenu();

begin

ShowMenuBar(['Чтение', 'Просмотр', 'Сортировка', 'Поиск', 'Добавить',

'Удалить', 'Редактировать', 'Подарки', 'Выход без сохранения',

'Выход с сохранением']);

end;

procedure ShowSweet;

begin

writeln(' ---------------------------------------------------------------------------- ');

writeln('|', sweet.info.id:7, ' |', sweet.info.name:30, ' |',

sweet.info.price:8, ' |', sweet.info.weight:7, ' |', sweet.info.sugar

:7, ' |');

writeln(' ---------------------------------------------------------------------------- ');

end;

procedure ShowTableHeader;

begin

// Interface size: id: 10, name: 34, price: 10, weight: 9, sugar: 9

writeln(' ---------------------------------------------------------------------------- ');

writeln('| ТИП |');

writeln('|----------------------------------------------------------------------------|');

writeln('| id | Наименование | Цена | Вес | Сахар |');

writeln('|----------------------------------------------------------------------------|');

end;

function SortManager;

var

response: string;

intVar: integer;

code: integer;

begin

Result := OK\_CODE;

ShowMenuBar([' По названию ', 'По цене', 'Назад']);

readln(response);

if TryStrToInt(response, intVar) then

begin

case intVar of

1:

code := CMP\_NAME;

2:

code := CMP\_PRICE;

3:

Result := OK\_CODE;

else

Result := INVALID\_INTEGER\_CODE;

end;

end

else

Result := INVALID\_VALUE\_CODE;

if Result = OK\_CODE then

begin

while groups^.next <> nil do

begin

groups := groups^.next;

SortSweets(groups^.sweets, code);

end;

if intVar <> 3 then

ShowMessage(1, 'Список сладостей успешно отсортирован!', 2);

end;

ShowMainMenu();

end;

procedure Start;

begin

New(groupToInit);

New(groupToInit.sweets);

New(presents);

New(presents.sweets);

groupToInit.info.id := 0;

groupToInit.sweets.info.id := 0;

if not(DirectoryExists('Sessions')) then

MkDir('Sessions');

PutCursorToPos(MAX\_WIDTH div 3, MAX\_HEIGHT div 3);

SetColor(clBlack, clYellow);

writeln(' СДЕЛАЙТЕ ПОЛНОЭКРАННЫЙ РЕЖИМ ');

PutCursorToPos(MAX\_WIDTH div 3, MAX\_HEIGHT div 3 + 1);

writeln('НАЖМИТЕ ENTER, ЧТОБЫ ЗАПУСТИТЬ БЕЛЫЙ ЭНДОРФИН ');

SetColor(clBlack, clWhite);

readln;

end;

var

groups: PGroup;

event: integer = 0;

sessionsExisting: TSearchRec;

tempTestGetGroupCombo: PGroupCombo;

begin

Start(groups);

ClearScreen(true);

ShowMainMenu();

while event <> EXIT\_CODE do

begin

event := KeyController(groups);

HandleEvent(event);

end;

Finish(groups);

readln;

end.

**Модуль FilesHandling:**

unit FilesHandling;

interface

uses

SysUtils,

windows,

System.Types,

System.IOUtils,

MonodirectionalList,

Include;

const

groupsFileName = 'groups.tgr';

sweetsFileName = 'sweets.tsw';

sweetSeparator: TSweetInfo = (id: -999;

name: '#0#0#0#0#0#0#0#0#0#0#0#0#0#0#0#0#0#0#0#0#0#0#0#0#0#0#0#0#0#0';);

var

SessionsPath: string = 'Sessions\';

isSessionOpen: boolean = false;

function OpenSession(groups: PGroup; folder: string): integer;

function SaveSession(groups: PGroup; folder: string): integer;

procedure ShowExistingSessions();

function GetExistingSessions(): TStringDynArray;

function isSessionExists(session: string): boolean;

implementation

type

TGroupsFile = File of TGroupInfo;

TSweetsFile = File of TSweetInfo;

var

groupsFile: TGroupsFile;

sweetsFile: TSweetsFile;

function GetExistingSessions;

var

temp: string;

begin

Result := TDirectory.GetDirectories(SessionsPath);

for var i := Low(Result) to High(Result) do

Result[i] := Copy(Result[i], Pos('\', Result[i]) + 1,

Length(Result[i]) - Pos('\', Result[i]));

end;

procedure ShowExistingSessions;

var

folders: TStringDynArray;

folder: string;

hOut: THandle;

k, i, index: integer;

begin

k := 1;

folders := TDirectory.GetDirectories(SessionsPath);

hOut := GetStdHandle(STD\_OUTPUT\_HANDLE);

writeln(' --- СПИСОК СУЩЕСТВУЮЩИХ СЕССИЙ --- ');

writeln;

for folder in folders do

begin

index := Pos(SessionsPath, folder);

index := index + Length(SessionsPath);

write(' ', k, '. ');

SetConsoleTextAttribute(hOut, clWhite);

for i := index to Length(folder) do

write(folder[i]);

SetConsoleTextAttribute(hOut, clLightGray);

k := k + 1;

writeln;

end;

writeln;

end;

function isSessionExists;

begin

if DirectoryExists(SessionsPath + session) then

Result := true

else

Result := false;

end;

function OpenSession;

var

temp: PGroup;

sweet: TSweetInfo;

sweets: PSweet;

begin

Result := OK\_CODE;

temp := groups;

if not(DirectoryExists(SessionsPath + folder)) then

Result := SESSION\_NOT\_FOUND\_CODE

else

begin

AssignFile(groupsFile, SessionsPath + folder + '\' + groupsFileName);

Reset(groupsFile);

AssignFile(sweetsFile, SessionsPath + folder + '\' + sweetsFileName);

Reset(sweetsFile);

read(groupsFile, groups.info);

read(sweetsFile, groups.sweets.info);

while not(Eof(groupsFile)) do

begin

New(groups.next);

groups := groups.next;

read(groupsFile, groups.info);

New(groups.sweets);

sweets := groups.sweets;

read(sweetsFile, sweet);

while (sweet.name <> sweetSeparator.name) do

begin

New(sweets.next);

sweets := sweets.next;

sweets.info := sweet;

read(sweetsFile, sweet);

end;

sweets.next := nil;

end;

CloseFile(groupsFile);

CloseFile(sweetsFile);

isSessionOpen := true;

end;

end;

function SaveSession;

var

filePath: string;

sweet: PSweet;

begin

Result := OK\_CODE;

if not(DirectoryExists(SessionsPath + folder)) then

MkDir(SessionsPath + folder);

AssignFile(groupsFile, SessionsPath + folder + '\' + groupsFileName);

AssignFile(sweetsFile, SessionsPath + folder + '\' + sweetsFileName);

Rewrite(groupsFile);

Rewrite(sweetsFile);

// For correct indexing work

write(groupsFile, groups.info);

write(sweetsFile, groups.sweets.info);

while groups.next <> nil do

begin

groups := groups.next;

write(groupsFile, groups.info);

sweet := groups.sweets;

while sweet.next <> nil do

begin

sweet := sweet.next;

write(sweetsFile, sweet.info);

end;

write(sweetsFile, sweetSeparator);

end;

CloseFile(groupsFile);

CloseFile(sweetsFile);

end;

end.

**Модуль MonodirectionalList:**

unit MonodirectionalList;

interface

const

// COMPARE OPTIONS

CMP\_ID = 1;

CMP\_NAME = 2;

CMP\_MAX\_PRICE = 3;

CMP\_MAX\_SUGAR\_PERCENT = 4;

CMP\_PRICE = 5;

CMP\_WEIGHT = 6;

CMP\_SUGAR = 7;

type

PSweet = ^TSweet;

PGroup = ^TGroup;

PPresent = ^TPresent;

TGroupInfo = record

id: integer;

name: string[30];

end;

TSweetInfo = record

id: integer;

name: string[30];

price: integer;

weight: integer;

sugar: integer;

end;

TPresentInfo = record

sweetType: string[30];

sweetName: string[30];

count: integer;

end;

TGroup = record

info: TGroupInfo;

sweets: PSweet;

next: PGroup;

end;

TSweet = record

info: TSweetInfo;

next: PSweet;

end;

TPresent = record

info: TPresentInfo;

next: PPresent;

end;

procedure AddGroup(groups: PGroup; name: string);

procedure AddSweet(groups: PGroup; id: integer; sweet: PSweet);

function CompareSweetFields(sweet1, sweet2: PSweet; field: integer): integer;

function isGroupExists(groups: PGroup; id: integer): boolean;

function isSweetExists(groups: PGroup; id: integer): boolean;

function ExtractSweetByID(groups: PGroup; id: integer): PSweet;

function DeleteGroup(groups: PGroup; idToDelete: integer): boolean;

function DeleteSweet(groups: PGroup; idToDelete: integer): boolean;

function GetGroupIDByName(groups: PGroup; name: string): integer;

function GetSweetIDByName(groups: PGroup; name: string): integer;

function GetGroupsCount(groups: PGroup): integer;

function GetSweetsCount(sweets: PSweet): integer;

procedure SortSweets(sweets: PSweet; code: integer);

implementation

uses

System.SysUtils;

procedure AddGroup(groups: PGroup; name: string);

var

id: integer;

begin

inc(groups.info.id);

id := groups.info.id;

while groups.next <> nil do

groups := groups^.next;

New(groups^.next);

groups := groups^.next;

groups.info.id := id;

groups.info.name := name;

groups^.next := nil;

New(groups^.sweets);

groups^.sweets.next := nil;

end;

procedure AddSweet(groups: PGroup; id: integer; sweet: PSweet);

var

sweets: PSweet;

temp: PGroup;

begin

temp := groups;

while groups^.next <> nil do

begin

groups := groups^.next;

if groups.info.id = id then

sweets := groups^.sweets;

end;

while sweets.next <> nil do

sweets := sweets.next;

New(sweets.next);

sweets := sweets.next;

inc(temp^.sweets.info.id);

sweets.info.id := temp^.sweets.info.id;

sweets.info.name := sweet.info.name;

sweets.info.price := sweet.info.price;

sweets.info.weight := sweet.info.weight;

sweets.info.sugar := sweet.info.sugar;

sweets.next := nil;

end;

function CompareSweetFields(sweet1, sweet2: PSweet; field: integer): integer;

var

a: integer;

begin

case field of

CMP\_ID:

Result := sweet1.info.id - sweet2.info.id;

CMP\_NAME:

begin

if LowerCase(sweet1.info.name, loUserLocale) <

LowerCase(sweet2.info.name, loUserLocale) then

Result := -1

else if LowerCase(sweet1.info.name, loUserLocale) >

LowerCase(sweet2.info.name, loUserLocale) then

Result := 1

else

Result := 0;

end;

CMP\_MAX\_PRICE:

if sweet1.info.price <= sweet2.info.price then

Result := 0

else

Result := -1;

CMP\_MAX\_SUGAR\_PERCENT:

if (sweet1.info.sugar / sweet1.info.weight) \* 100 < sweet2.info.sugar then

Result := 0

else

Result := -1;

CMP\_PRICE:

Result := sweet1.info.price - sweet2.info.price;

CMP\_WEIGHT:

Result := sweet1.info.weight - sweet2.info.weight;

CMP\_SUGAR:

Result := sweet1.info.sugar - sweet2.info.sugar;

else

Result := -999999;

end;

end;

function DeleteGroup(groups: PGroup; idToDelete: integer): boolean;

var

temp: PGroup;

currSweet: PSweet;

tempSweet: PSweet;

begin

Result := false;

while ((groups^.next <> nil) and not(Result)) do

begin

if groups.next.info.id = idToDelete then

begin

temp := groups^.next;

groups^.next := groups^.next^.next;

Result := true;

currSweet := temp.sweets;

while currSweet.next <> nil do

begin

tempSweet := currSweet.next;

currSweet.next := currSweet.next.next;

tempSweet.next := nil;

Dispose(tempSweet);

end;

// NilAndDispose

temp.sweets := nil;

temp.next := nil;

Dispose(temp.sweets);

Dispose(temp);

end;

// In case of last element deletion

if groups^.next <> nil then

groups := groups^.next;

end;

end;

function DeleteSweet(groups: PGroup; idToDelete: integer): boolean;

var

temp: PSweet;

sweets: PSweet;

begin

Result := false;

while ((groups^.next <> nil) and not(Result)) do

begin

groups := groups^.next;

sweets := groups^.sweets;

while sweets.next <> nil do

begin

if sweets.next.info.id = idToDelete then

begin

temp := sweets.next;

sweets.next := sweets.next^.next;

Result := true;

temp.next := nil;

Dispose(temp);

end;

if sweets.next <> nil then

sweets := sweets.next;

end;

end;

end;

function ExtractSweetByID(groups: PGroup; id: integer): PSweet;

var

sweets: PSweet;

begin

New(Result);

Result^.info.id := -1; // id = -1 means sweet not found

while ((groups.next <> nil) and (Result^.info.id = -1)) do

begin

groups := groups^.next;

sweets := groups^.sweets;

while ((sweets.next <> nil) and (Result^.info.id = -1)) do

begin

sweets := sweets.next;

if sweets.info.id = id then

begin

Dispose(Result);

Result := sweets;

end;

end;

end;

end;

function GetGroupsCount(groups: PGroup): integer;

begin

Result := 0;

while groups.next <> nil do

begin

groups := groups.next;

inc(Result);

end;

end;

function GetGroupIDByName(groups: PGroup; name: string): integer;

begin

Result := -1;

name := LowerCase(name, loUserLocale);

while groups^.next <> nil do

begin

groups := groups^.next;

if LowerCase(groups.info.name, loUserLocale) = name then

Result := groups.info.id;

end;

end;

function GetSweetIDByName(groups: PGroup; name: string): integer;

var

sweets: PSweet;

begin

Result := -1;

name := LowerCase(name, loUserLocale);

while ((groups.next <> nil) and (Result = -1)) do

begin

groups := groups^.next;

sweets := groups^.sweets;

while ((sweets.next <> nil) and (Result = -1)) do

begin

sweets := sweets.next;

if LowerCase(sweets.info.name, loUserLocale) = name then

Result := sweets.info.id;

end;

end;

end;

function GetSweetsCount(sweets: PSweet): integer;

begin

Result := 0;

while sweets.next <> nil do

begin

sweets := sweets.next;

inc(Result);

end;

end;

function isGroupExists(groups: PGroup; id: integer): boolean;

begin

Result := false;

while (groups.next <> nil) and not(Result) do

begin

groups := groups^.next;

if groups.info.id = id then

Result := true;

end;

end;

function isSweetExists(groups: PGroup; id: integer): boolean;

var

sweets: PSweet;

begin

Result := false;

while (groups.next <> nil) and not(Result) do

begin

groups := groups^.next;

sweets := groups^.sweets;

while (sweets.next <> nil) and not(Result) do

begin

sweets := sweets.next;

if sweets.info.id = id then

Result := true;

end;

end;

end;

procedure SortSweets;

var

minPSweet, currPSweet, tempSweet: PSweet;

begin

currPSweet := sweets;

New(tempSweet);

while currPSweet.next <> nil do

begin

currPSweet := currPSweet.next;

minPSweet := currPSweet;

sweets := currPSweet;

while sweets.next <> nil do

begin

sweets := sweets.next;

if CompareSweetFields(sweets, minPSweet, code) < 0 then

minPSweet := sweets;

end;

// May be this piece is should be rewritted as Swap procedure

tempSweet.info.id := currPSweet.info.id;

tempSweet.info.name := currPSweet.info.name;

tempSweet.info.price := currPSweet.info.price;

tempSweet.info.weight := currPSweet.info.weight;

tempSweet.info.sugar := currPSweet.info.sugar;

currPSweet.info.id := minPSweet.info.id;

currPSweet.info.name := minPSweet.info.name;

currPSweet.info.price := minPSweet.info.price;

currPSweet.info.weight := minPSweet.info.weight;

currPSweet.info.sugar := minPSweet.info.sugar;

minPSweet.info.id := tempSweet.info.id;

minPSweet.info.name := tempSweet.info.name;

minPSweet.info.price := tempSweet.info.price;

minPSweet.info.weight := tempSweet.info.weight;

minPSweet.info.sugar := tempSweet.info.sugar;

end;

end;

end.

**Модуль Include:**

unit Include;

interface

const

MAX\_WIDTH = 160;

MAX\_HEIGHT = 60;

INPUT\_LINE = 1;

// CODES

OK\_CODE = 0;

EXIT\_CODE = -1;

INVALID\_VALUE\_CODE = 1;

INVALID\_INTEGER\_CODE = 2;

INVALID\_PERCENT\_CODE = 3;

FEATURE\_NOT\_EXISTS\_CODE = 4;

UNKNOWN\_ERROR\_CODE = 5;

SESSION\_NOT\_FOUND\_CODE = 6;

LENGTH\_EXCEED\_CODE = 7;

INVALID\_INPUT\_CODE = 8;

// COLORS

clBlack = 0;

clBlue = 1;

clGreen = 2;

clCyan = 3;

clRed = 4;

clMagenta = 5;

clBrown = 6;

clLightGray = 7;

clDarkGray = 8;

clLightBlue = 9;

clLightGreen = 10;

clLightCyan = 11;

clLightRed = 12;

clLightMagenta = 13;

clYellow = 14;

clWhite = 15;

// TEMPORARY ARRAYS WITH VALUES TO INITIALIZE OUR LISTS

var

groupNames: array of string = ['Конфеты', 'Шоколад', 'Печенье', 'Торты'];

sweetNames: array of array of string =

[['Злая Пчёлка', 'Ромашка', 'Арабская ночь', 'Сладкий Рог', 'Raffaello'],

['Любимая Алёнка', 'Президент 56%', 'Max Fun', 'Milka',

'Спартак Premium'], ['Слодыч Васiльковы', 'Oreo Strawberry', 'Яшкино'],

['Наполеон', 'Бонапарт']];

sweetPrices: array of array of integer = [[10, 15, 20, 30, 120],

[220, 250, 620, 490, 310], [90, 300, 150], [1990, 2199]];

sweetWeights: array of array of integer = [[7, 15, 15, 12, 10],

[100, 100, 200, 85, 95], [200, 120, 300], [500, 700]];

sweetSugars: array of array of integer = [[2, 8, 6, 6, 2],

[40, 25, 85, 43, 30], [75, 65, 150], [200, 350]];

implementation

end.

**Модуль GiftGenerator:**

unit GiftGenerator;

interface

uses

MonodirectionalList,

System.SysUtils;

type

PGroupCombo = ^TGroupCombo;

PSurprizeSweet = ^TSurprizeSweet;

PSurprize = ^TSurprize;

TGroupCombo = record

group: PGroup;

next: PGroupCombo;

end;

TSurprizeInfo = record

name: string[30];

price: integer;

weight: integer;

sugar: integer;

end;

TSurprizeSweetInfo = record

name: string[30];

count: integer;

end;

TSurprizeSweet = record

info: TSurprizeSweetInfo;

next: PSurprizeSweet;

end;

TSurprize = record

info: TSurprizeInfo;

sweets: PSurprizeSweet;

next: PSurprize;

end;

const

MAX\_SWEETS\_COUNT = 25;

SurpTemplate = 'ПОДАРОК ';

MAX\_PRESENTS\_COUNT = 40000;

var

fileName: string = 'suprizes.txt';

PresentsPath: string = 'Presents\';

GlobalCount: integer = 1;

presentsOverflow: boolean;

procedure BitmaskInc(var bitMask: array of integer; len: integer);

function Factorial(n: integer): Int64;

function Power2(n: integer): integer;

function GetCombinationsCount(n, k: integer): integer;

function GetGroupsCombinations(groups: PGroup; k: integer): PGroupCombo;

function Get1Count(arr: array of integer): integer;

procedure ShowPresentCombos(combos: PGroupCombo; k: integer);

procedure ShowPresent(presentToShow: PSurprize);

procedure SavePresents(presentToSave: PSurprize; fileToSave: string);

procedure SortPresents(presentsToSort: PSurprize);

procedure DisposePresents(presentsToDispose: PSurprize);

function CreatePresents(var currPresent: PSurprize; sweets: array of PSweet;

bitMask: array of integer; maxPrice, maxWeight: integer): PSurprize;

function MainHandler(groups: PGroup; maxPrice, maxWeight, typesCount: integer)

: PSurprize;

implementation

function ArraySum(arr: array of integer): integer; forward;

function isValidBitmask(bitMask: array of integer;

groupVolumes: array of integer): boolean; forward;

function ArraySum;

begin

Result := 0;

for var i := Low(arr) to High(arr) do

Result := Result + arr[i];

end;

procedure BitmaskInc;

// Увеличивает битмаску на единицу (по аналогии с числом в двоичном виде)

// Считается, что на вход не будет подан массив вида [1, 1, 1, 1, 1], иначе функция отработает некорректно

var

flag: boolean;

begin

flag := True;

Dec(len); // Cause dynamic array is 0..len-1

if bitMask[len] = 0 then

begin

bitMask[len] := 1;

flag := False;

end

else

bitMask[len] := 0;

while flag do

begin

len := len - 1;

if bitMask[len] = 0 then

begin

bitMask[len] := 1;

flag := False;

end

else

bitMask[len] := 0;

end;

end;

function CreatePresents;

var

activeSweets: array of PSweet;

activeSweetsLen: integer;

i, len: integer;

presents, presentsPrev: PSurprize;

currPresentSweet, prevPresentSweet: PSurprizeSweet;

function PresentsRecursive(currPresentSweet: PSurprizeSweet;

len, maxPrice, maxWeight: integer): boolean;

var

n, m, count: integer;

begin

Result := False;

if len = 1 then

begin

n := maxPrice div activeSweets[len - 1].info.price;

m := maxWeight div activeSweets[len - 1].info.weight;

if n < m then

count := n

else

count := m;

if count > 0 then

begin

Result := True;

inc(GlobalCount);

if GlobalCount > MAX\_PRESENTS\_COUNT then

presentsOverflow := True;

New(currPresentSweet.next);

currPresentSweet := currPresentSweet.next;

currPresentSweet.next := nil;

currPresentSweet.info.name := activeSweets[len - 1].info.name;

currPresentSweet.info.count := count;

currPresent.info.price := currPresent.info.price + count \* activeSweets

[len - 1].info.price;

currPresent.info.weight := currPresent.info.weight + count \*

activeSweets[len - 1].info.weight;

currPresent.info.sugar := currPresent.info.sugar + count \* activeSweets

[len - 1].info.sugar;

end

else

begin

currPresentSweet := currPresent.sweets;

while currPresentSweet.next <> nil do

begin

prevPresentSweet := currPresentSweet;

currPresentSweet := currPresentSweet.next;

Dispose(prevPresentSweet);

end;

Dispose(currPresentSweet);

currPresent := presentsPrev;

Dispose(currPresent.next);

currPresent.next := nil;

Result := False;

end;

end

else

begin

n := 1;

while (n \* activeSweets[len - 1].info.price <= maxPrice) and

(n \* activeSweets[len - 1].info.weight <= maxWeight) and

not(presentsOverflow) do

begin

New(currPresentSweet.next);

currPresentSweet := currPresentSweet.next;

currPresentSweet.next := nil;

currPresentSweet.info.name := activeSweets[len - 1].info.name;

currPresentSweet.info.count := n;

currPresent.info.price := currPresent.info.price + n \* activeSweets

[len - 1].info.price;

currPresent.info.weight := currPresent.info.weight + n \* activeSweets

[len - 1].info.weight;

currPresent.info.sugar := currPresent.info.sugar + n \* activeSweets

[len - 1].info.sugar;

if PresentsRecursive(currPresentSweet, len - 1,

maxPrice - n \* activeSweets[len - 1].info.price,

maxWeight - n \* activeSweets[len - 1].info.weight) then

begin

Result := True;

inc(n);

if (n \* activeSweets[len - 1].info.price <= maxPrice) and

(n \* activeSweets[len - 1].info.weight <= maxWeight) and

not(presentsOverflow) then

begin

presentsPrev := currPresent;

New(currPresent.next);

currPresent := currPresent.next;

New(currPresent.sweets);

currPresent.sweets.next := nil;

currPresent.info.price := 0;

currPresent.info.weight := 0;

currPresent.info.sugar := 0;

currPresent.info.name := SurpTemplate + IntToStr(GlobalCount);

prevPresentSweet := presentsPrev.sweets;

currPresentSweet := currPresent.sweets;

m := activeSweetsLen;

while (m > len) do

begin

New(currPresentSweet.next);

currPresentSweet := currPresentSweet.next;

currPresentSweet.next := nil;

prevPresentSweet := prevPresentSweet.next;

currPresentSweet.info := prevPresentSweet.info;

currPresent.info.price := currPresent.info.price +

prevPresentSweet.info.count \* activeSweets[m - 1].info.price;

currPresent.info.weight := currPresent.info.weight +

prevPresentSweet.info.count \* activeSweets[m - 1].info.weight;

currPresent.info.sugar := currPresent.info.sugar +

prevPresentSweet.info.count \* activeSweets[m - 1].info.sugar;

Dec(m);

end;

end;

end

else

begin

inc(n);

break;

end;

end;

if n = 1 then

begin

currPresentSweet := currPresent.sweets;

while currPresentSweet.next <> nil do

begin

prevPresentSweet := currPresentSweet;

currPresentSweet := currPresentSweet.next;

Dispose(prevPresentSweet);

end;

Dispose(currPresentSweet);

currPresent := presentsPrev;

Dispose(currPresent.next);

currPresent.next := nil;

Result := False;

end;

end;

end;

begin

presentsPrev := currPresent;

New(currPresent.next);

currPresent := currPresent.next;

currPresent.info.name := SurpTemplate + IntToStr(GlobalCount);

currPresent.info.price := 0;

currPresent.info.weight := 0;

currPresent.info.sugar := 0;

New(currPresent.sweets);

currPresentSweet := currPresent.sweets;

currPresentSweet.next := nil;

SetLength(activeSweets, MAX\_SWEETS\_COUNT);

len := 0;

for i := Low(bitMask) to High(bitMask) do

begin

if bitMask[i] = 1 then

begin

activeSweets[len] := sweets[i];

inc(len);

end;

end;

activeSweetsLen := len;

PresentsRecursive(currPresentSweet, len, maxPrice, maxWeight);

end;

procedure DisposePresents(presentsToDispose: PSurprize);

var

curr, prev: PSurprizeSweet;

prevPresent: PSurprize;

begin

while presentsToDispose.next <> nil do

begin

prevPresent := presentsToDispose;

presentsToDispose := presentsToDispose.next;

curr := presentsToDispose.sweets;

while curr.next <> nil do

begin

prev := curr;

curr := curr.next;

prev.next := nil;

Dispose(prev);

end;

curr.next := nil;

prevPresent.sweets := nil;

prevPresent.next := nil;

Dispose(curr);

Dispose(prevPresent.sweets);

Dispose(prevPresent);

end;

presentsToDispose.next := nil;

Dispose(presentsToDispose);

end;

function Factorial;

begin

Result := 1;

for var i := n downto 1 do

begin

Result := Result \* i;

end;

end;

function GetCombinationsCount;

begin

if (k > n) or (k < 0) or (n < 0) then

Result := -1

else

Result := Factorial(n) div (Factorial(n - k) \* Factorial(k));

end;

function Get1Count;

begin

Result := 0;

for var i := Low(arr) to High(arr) do

if arr[i] = 1 then

inc(Result);

end;

function GetGroupsCombinations;

var

resultEdge: PGroupCombo;

groupsArray: array of PGroup;

groupsCount, i, j: integer;

iterations: integer;

bitMask: array of integer;

ch: char;

begin

New(Result);

resultEdge := Result;

SetLength(groupsArray, GetGroupsCount(groups));

groupsCount := 0;

while groups.next <> nil do

begin

groups := groups.next;

groupsArray[groupsCount] := groups;

inc(groupsCount);

end;

iterations := Power2(groupsCount) - 1;

SetLength(bitMask, groupsCount);

for i := Low(bitMask) to High(bitMask) do

bitMask[i] := 0;

for i := 1 to iterations do

begin

BitmaskInc(bitMask, groupsCount);

if Get1Count(bitMask) = k then

begin

for j := Low(bitMask) to High(bitMask) do

begin

if bitMask[j] = 1 then

begin

New(resultEdge.next);

resultEdge := resultEdge.next;

resultEdge.group := groupsArray[j];

end;

end;

end;

end;

end;

function isValidBitmask;

var

isGroupPresent: boolean;

k, i, currGroup: integer;

begin

Result := False;

isGroupPresent := True;

k := Low(bitMask); // = 0

currGroup := Low(groupVolumes);

while (k < Length(bitMask)) and isGroupPresent do

begin

isGroupPresent := False;

for i := k to k + groupVolumes[currGroup] - 1 do

begin

if bitMask[i] = 1 then

isGroupPresent := True;

end;

k := k + groupVolumes[currGroup];

inc(currGroup);

end;

if isGroupPresent then

Result := True;

end;

function MainHandler;

var

sweetsArray: array of PSweet;

sweetsLen: integer;

currSweet: PSweet;

groupVolumes: array of integer;

edgePresent: PSurprize;

bitMask: array of integer;

bitMaskLen: integer;

iterations: integer;

groupCombos: PGroupCombo;

edgeCombo: PGroupCombo;

comboCount, groupsCount, i, j, k: integer;

begin

presentsOverflow := False;

GlobalCount := 0;

New(Result);

edgePresent := Result;

groupsCount := GetGroupsCount(groups);

SetLength(groupVolumes, typesCount);

groupCombos := GetGroupsCombinations(groups, typesCount);

edgeCombo := groupCombos.next;

comboCount := GetCombinationsCount(groupsCount, typesCount);

// ShowPresentCombos(groupCombos, typesCount);

// 25 because combinations of more than 20 sweets are just impossible

// Maybe there are impossible with even less sweets

SetLength(sweetsArray, MAX\_SWEETS\_COUNT);

k := 0;

while (edgeCombo <> nil) and not(presentsOverflow) do

begin

sweetsLen := 0;

for i := Low(groupVolumes) to High(groupVolumes) do

begin

currSweet := edgeCombo.group.sweets;

groupVolumes[i] := GetSweetsCount(currSweet);

while currSweet.next <> nil do

begin

currSweet := currSweet.next;

sweetsArray[sweetsLen] := currSweet;

inc(sweetsLen);

end;

edgeCombo := edgeCombo.next;

end;

bitMaskLen := ArraySum(groupVolumes);

SetLength(bitMask, bitMaskLen);

for i := Low(bitMask) to High(bitMask) do

bitMask[i] := 0;

iterations := Power2(bitMaskLen) - 1;

i := 1;

while (i <= iterations) and not(presentsOverflow) do

begin

if GlobalCount > MAX\_PRESENTS\_COUNT then

presentsOverflow := True

else

begin

BitmaskInc(bitMask, bitMaskLen);

if isValidBitmask(bitMask, groupVolumes) then

CreatePresents(edgePresent, sweetsArray, bitMask, maxPrice,

maxWeight);

end;

inc(i);

end;

end;

end;

function Power2;

begin

Result := 1;

for var i := 1 to n do

Result := Result \* 2;

end;

procedure SavePresents;

var

FFile: TextFile;

currSweet: PSurprizeSweet;

begin

Assign(FFile, fileToSave);

Rewrite(FFile);

while presentToSave.next <> nil do

begin

presentToSave := presentToSave.next;

// writeln(FFile, '');

writeln(FFile, ' ------------------------------------------------ ');

writeln(FFile, '| ', presentToSave.info.name:40, ' |');

writeln(FFile, ' ------------------------------------------------ ');

writeln(FFile, '| ЦЕНА : ', presentToSave.info.price:30, ' |');

writeln(FFile, '| ВЕС : ', presentToSave.info.weight:30, ' |');

writeln(FFile, '| CАХАР: ', presentToSave.info.sugar:30, ' |');

writeln(FFile, ' ------------------------------------------------ ');

writeln(FFile, '| Сладости внутри: |');

writeln(FFile, '| | |');

currSweet := presentToSave.sweets;

while currSweet.next <> nil do

begin

currSweet := currSweet.next;

writeln(FFile, '| ', currSweet.info.name:33, ' | ', currSweet.info.count

:10, ' |');

end;

writeln(FFile, '| | |');

writeln(FFile, ' ------------------------------------------------ ');

end;

Close(FFile);

end;

procedure ShowPresent;

var

currSweet: PSurprizeSweet;

begin

writeln;

writeln(' ------------------------------------------------ ');

writeln('| ', presentToShow.info.name:40, ' |');

writeln(' ------------------------------------------------ ');

writeln('| ЦЕНА : ', presentToShow.info.price:30, ' |');

writeln('| ВЕС : ', presentToShow.info.weight:30, ' |');

writeln('| CАХАР: ', presentToShow.info.sugar:30, ' |');

writeln(' ------------------------------------------------ ');

writeln('| Сладости внутри: |');

writeln('| | |');

currSweet := presentToShow.sweets;

while currSweet.next <> nil do

begin

currSweet := currSweet.next;

writeln('| ', currSweet.info.name:33, ' | ', currSweet.info.count:10, ' |');

end;

writeln('| | |');

writeln(' ------------------------------------------------ ');

end;

procedure ShowPresentCombos;

var

count: integer;

begin

count := 0;

while combos.next <> nil do

begin

combos := combos.next;

write(combos.group.info.name, ' ');

inc(count);

if count mod k = 0 then

writeln;

end;

end;

procedure SortPresents;

var

minPresent, currPresent, tempPresent: PSurprize;

begin

currPresent := presentsToSort;

New(tempPresent);

while currPresent.next <> nil do

begin

currPresent := currPresent.next;

minPresent := currPresent;

presentsToSort := currPresent;

while presentsToSort.next <> nil do

begin

presentsToSort := presentsToSort.next;

if presentsToSort.info.sugar < minPresent.info.sugar then

minPresent := presentsToSort;

end;

tempPresent.info.name := currPresent.info.name;

tempPresent.info.price := currPresent.info.price;

tempPresent.info.weight := currPresent.info.weight;

tempPresent.info.sugar := currPresent.info.sugar;

currPresent.info.name := minPresent.info.name;

currPresent.info.price := minPresent.info.price;

currPresent.info.weight := minPresent.info.weight;

currPresent.info.sugar := minPresent.info.sugar;

minPresent.info.name := tempPresent.info.name;

minPresent.info.price := tempPresent.info.price;

minPresent.info.weight := tempPresent.info.weight;

minPresent.info.sugar := tempPresent.info.sugar;

end;

end;

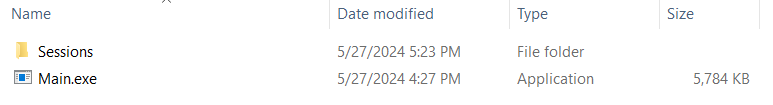
end.

**Белый Эндорфин - руководство по эксплуатации**

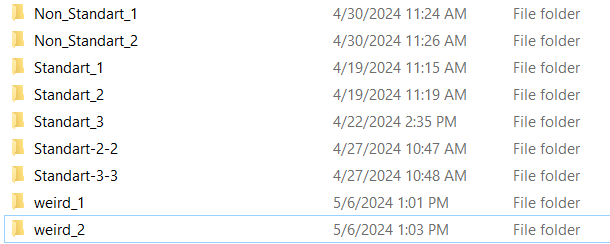
# 1. Установка программы

Для установки программы необходимо скачать исполняемый файл *main.exe* Далее файл помещается в удобное место. Во время использования программы, рядом с исполняемым файлом cоздается папка Sessions, в которой хранятся все сохраненные сессии.

Правильная структура корневого каталога программы:



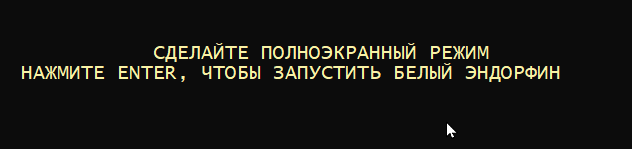
Возможная структура папки Sessions (в каждой папке два файла: groups.tgr и sweets.tsw):



# 2. Эксплуатация

Чтобы начать использовать программу необходимо запустить ее, открыв main.exe

Пользователю будет предложено перейти в полноэкранный режим и запустить программу:

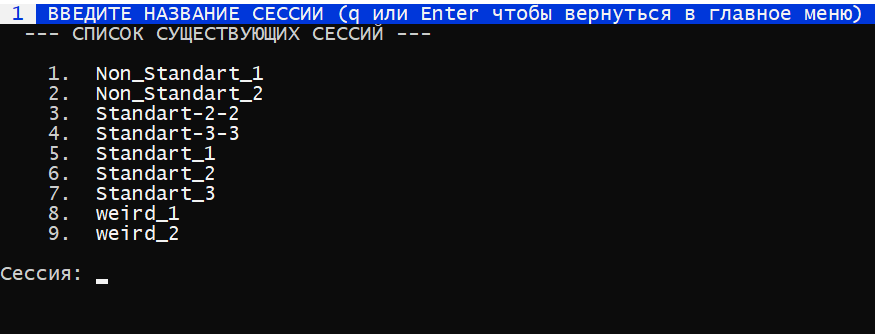


Меню программы состоит из 10 пунктов. Выбор пункта меню и все взаимодействия с программой производятся путем ввода номера действия и нажатия клавиши Enter.

**Описание основных функций**

## 1. Чтение данных из файла

Программа предложит список ранее созданных сессий, которые можно открыть, введя их номер или название:



## 2. Просмотр всего списка

Для вывода списка, необходимо выбрать один из предложенных списков. На экран будут выведены все элементы в том порядке, в котором они хранятся в данном списке. Результат работы функции на примере списка сладостей:

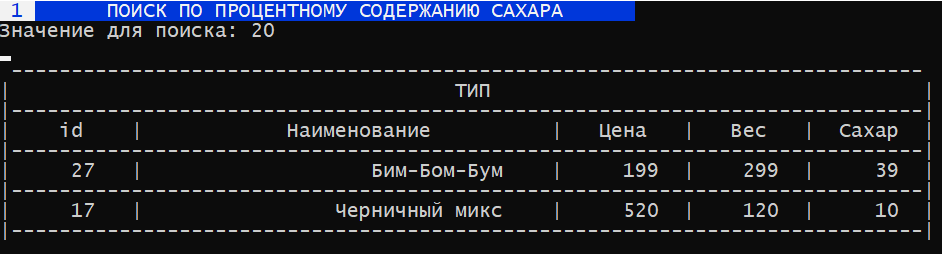


## 3. Сортировка сладостей

Для сортировки сладостей пользователь должен выбрать поле, по которому будет происходить сортировка. В результате список будет отсортирован по возрастанию выбранного поля. Поля для сортировки: Название, Цена.

## 4. Поиск сладостей с использованием фильтров

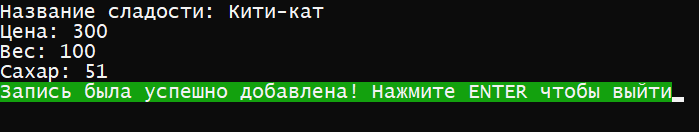
Для поиска сладостей пользователь должен выбрать поле, по которому будет произведен поиск и значение этого поля. В результате программа выводит список всех попавших под фильтр записей:



## 5. Добавление данных

Для добавления данных в определенный список, необходимо выбрать список (группы или сладости) и заполнить все поля для новой позиции. Если поле введено неверно, то выскочит соответствующее сообщение и пользователю будет предложено заново ввести значение этого поля.

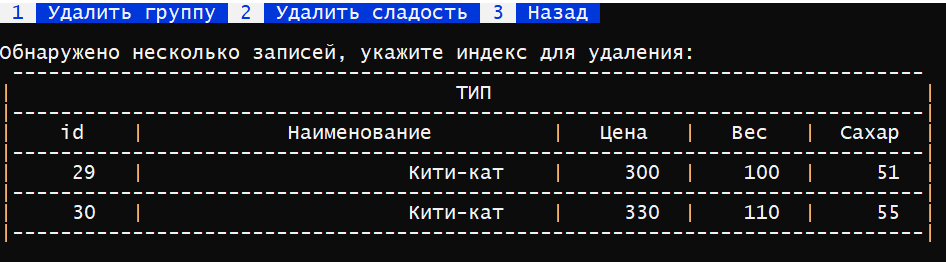
Пример добавления новой сладости:



## 6. Удаление данных их списков

Для удаления необходимо выбрать список. Затем ввести название или ID позиции к удалению. Если найдено несколько совпадений, то пользователю предлагается выбрать ID удаляемой позиции из совпавших. В случае удаления группы удаляются и все сладости в этой группе.

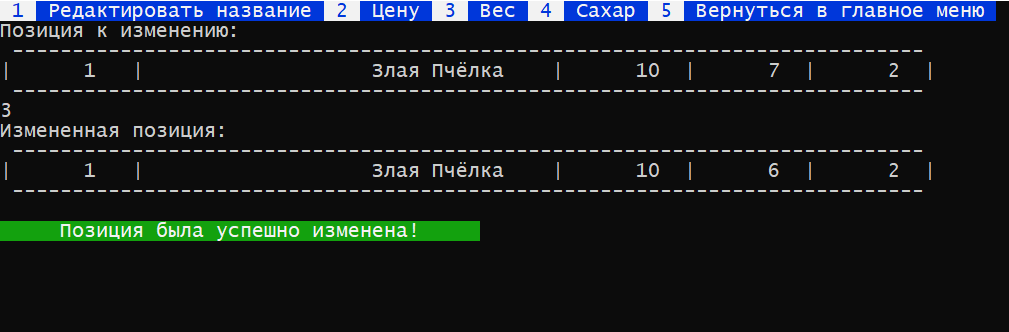
Пример совпадения по имени:



## 7. Редактирование сладостей

Для редактирования сладости, необходимо ввести ее Название или ID. В случае, если несколько элементов найдено, то предлагается выбрать конкретный. Затем необходимо выбрать поле для редактирования и ввести новое значение.

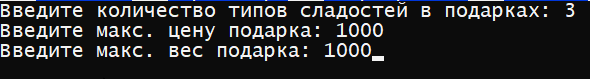
Пример:



## 8. Генерация подарков

При выборе данной функции, предлагается ввести ограничения для подарков, после ввода данных произойдет генерация подарков в файл surprizes.txt (находится в корневом каталоге программы. Количество подарков ограниченно 40001 штуками)

Пример ввода ограничения:



## 9. Выход из программы без сохранения изменений

Все действия сделанные в программе никуда не сохраняются и производится выход из программы. Будьте осторожны!

## 10. Выход с сохранением изменений

Все созданные списки будут сохранены в сессию под введенным названием.

По ходу работы с программой появляются информационные сообщения, которые указывают пользователю на ошибки, совершенные действия и результаты работы использованных функций.

Спасибо за использование **Белого Эндорфина**!

**ПРИЛОЖЕНИЕ А**

**(обязательное)**

**Разработка схемы программы**



Рисунок А.1 – Главное тело программы



Рисунок А.2 – Менеджер вывода



Рисунок А.3 – Специальная функция генерации подарков

**ПРИЛОЖЕНИЕ Б**

**(обязательное)**

**Техническое задание**

|  |  |
| --- | --- |
| **УТВЕРЖДАЮ**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ г. |  |
| Программное средство учета кондитерских изделий  Белый Эндорфин  Техническое задание  Действует с 10.03.2024 | |
| **СОГЛАСОВАНО**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г. | **РАЗРАБОТЧИК**  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г. |
| Минск, 2024 | |

Содержание

1.ОБЩИЕ СВЕДЕНИЯ 71

Полное наименование Системы и её условное обозначение 71

1.1. Наименование и реквизиты Заказчика 71

1.2. Наименование и реквизиты Исполнителя 71

1.3. Основание для разработки 71

1.4. Плановые сроки начала и окончания работ 71

2.НАЗНАЧЕНИЕ И ЦЕЛИ СОЗДАНИЯ СИСТЕМЫ 72

2.1. Назначение Системы 72

2.2. Цели создания Системы 72

3.ХАРАКТЕРИСТИКА ОБЪЕКТА АВТОМАТИЗАЦИИ 73

3.1. Краткие сведения об объектах автоматизации 73

3.2. Сведения об условиях эксплуатации объекта автоматизации и характеристиках окружающей среды 73

4.ТРЕБОВАНИЯ К СИСТЕМЕ 74

4.1. Требования к системе в целом 74

4.1.1.Требования к структуре и функционированию 74

4.1.1.1. Перечень подсистем, их назначение и основные характеристики 74

4.1.1.2. Требования к организации обмена информацией между компонентами Системы 74

4.1.1.3. Перспективы развития, модернизации Системы 74

4.2. Функциональные требования 74

4.3. Обработка ошибок 75

4.3.1.Ошибки загрузки данных из внешних источников 75

4.3.2.Внутренние ошибки 75

4.4. Интерфейс 75

4.4.1.Основные требования 75

4.4.1.1. Представление форм ввода данных 76

4.5. Требования к видам обеспечения 77

4.5.1.Требования к аппаратному обеспечению 77

4.5.2.Требования к программному обеспечению 77

4.5.3.Требования к лингвистическому обеспечению 77

4.5.3.1. Требования к языкам и средствам манипулирования данными………………………………………………………………………..77

# ОБЩИЕ СВЕДЕНИЯ

## Полное наименование Системы и её условное обозначение

Программное средство учета кондитерских изделий **Белый Эндорфин**

## Наименование и реквизиты Заказчика

Заказчиком Системы является кафедра ПОИТ (ассистент Фадеева Елена Евгеньевна)

Контактный телефон: +375 44 552 37 38

## Наименование и реквизиты Исполнителя

Исполнителем является (проектный менеджер – Будников Михаил Алексеевич)

Адрес электронной почты: Merlin\_Study@proton.me

Телефон: +375 44 535 06 16

## Основание для разработки

* Список индивидуальных заданий УП
* Список вариантов к списку индивидуальных заданий

## Плановые сроки начала и окончания работ

Начало разработки: 10 марта 2024 года;

Ожидаемое завершение разработки: 31 мая 2024 года

# НАЗНАЧЕНИЕ И ЦЕЛИ СОЗДАНИЯ СИСТЕМЫ

## Назначение Системы

**«Белый Эндорфин»** представляет собой систему для учета кондитерских изделий и составления персональных новогодних подарков по заданным критериям. Отхватите свою дозу сладкого удовольствия! Возможности:

* Просмотр таблицы сладостей, их цены, массы, количества сахара
* Поиск позиций по заданным параметрам
* Добавление позиций
* Редактирование отдельных позиций
* Удаление позиций
* Составление персонального новогоднего подарка по следующим характеристикам: вес подарка, количество типов сладостей и общей стоимости.
* Сортировка сформированных подарков по количеству сахара
* Сохранение списков в соответствующие файлы

## Цели создания Системы

Целями создания системы являются:

* Автоматизация формирования подарков.
* Автоматизация операций добавления, редактирования и удаления данных о кондитерских изделиях.
* Цифровизация данных о кондитерских изделиях и сладостях.
* Повышение информированности пользователей о количестве сахара в составе конкретных кондитерских изделиий и сладостей.

# ХАРАКТЕРИСТИКА ОБЪЕКТА АВТОМАТИЗАЦИИ

## Краткие сведения об объектах автоматизации

В программе организованы следующие объекты автоматизации:

1. Список групп сладостей. Состоит из следующих полей:
2. Код группы
3. Наименование группы
4. Ссылка на список соответствующих сладостей (внутреннее)
5. Ссылка на следующую группу (внутреннее)
6. Список сладостей. Состоит из следующих полей:
7. Код сладости
8. Наименование сладости
9. Цена
10. Вес
11. Сахар
12. Ссылка на следующую сладость (внутреннее)
13. Список сформированных подарков
14. Код подарка
15. Название группы подарков, сформированных по введенному критерию (Опциональный параметр, требует уточнения).
16. Количество типов сладостей
17. Цена подарка
18. Вес подарка
19. Ссылка на список сладостей в подарке (внутренний)
20. Ссылка на следующий подарок (внутренний)
21. Список сладостей в подарке
22. Тип сладости
23. Код сладости (внутренний, опциональный, если будет необходимо использование функций, завязанных на доступе по id, то он будет внедрен)
24. Название сладости
25. Количество штук.
26. Ссылка на следующую сладость (внутренний)

Внутренние поля не видны конечному пользователю.

## Сведения об условиях эксплуатации объекта автоматизации и характеристиках окружающей среды

Проект планируется использовать в течение года и хранить несколько тысяч торговых позиций.

# ТРЕБОВАНИЯ К СИСТЕМЕ

## Требования к системе в целом

### Требования к структуре и функционированию

#### Перечень подсистем, их назначение и основные характеристики

1. Просмотр списков.
2. Сортировка данных в соответствии с заданием.
3. Поиск данных по различным параметрам.
4. Добавление, удаление и редактирование данных из списка.
5. Формирование подарков из сладостей по критериям.

#### Требования к организации обмена информацией между компонентами Системы

• Отображение списка должно проходить путем чтения данных из списка.

• Обмен информацией между подсистемами должен происходить путем передачи информации из файлов, если это чтение данных, или путем передачи из списков, в других случаях.

#### Перспективы развития, модернизации Системы

В перспективе должен быть улучшен пользовательский интерфейс, путем перехода программного средства на Delphi VCL Form (Графический интерфейс).

## Функциональные требования

1. Подсистема формирования новогоднего подарка

Задачи:

А) Получение данных от пользователя

Б) Проверка введенных данных на корректность;

В) Вызов соответствующей подпрограммы для формирования подарков;

Г) Сохранение сформированных подарков в соответствующие списки

1. Подсистема поиска сладости по заданным параметрам

Задачи:

А) Обработка полученных данных.

Б) Поиск в списке сладостей записи, соответствующей параметрам, полученным от пользователя.

1. Подсистема сортировки

Задачи:

А) Сортировка подарков по возрастанию суммарного содержания сахара

1. Подсистема добавления/редактирования/удаления записи из списка

Задачи:

А) Получение от пользователя данных об операции и торговой позиции. Вывод ошибки при некорректных введенных данных.

Б) Добавление/редактирование/удаление выбранной пользователем записи из списка.

1. Подсистема просмотра списка:

Задачи:

А) Получение от пользователя номера списка для вывода

Б) Отображение соответствующего списка (Предупреждение при неверных введенных данных)

## Обработка ошибок

### Ошибки загрузки данных из внешних источников

Ошибки загрузки данных из внешних источников могут возникать при:

* Неверном формате файла-источника;
* Попытке загрузки данных из поврежденного файла;
* Отсутствии искомого файла;

При возникновении ошибок необходимо оповестить пользователя и прекратить работу подпрограммы, в которой произошла ошибка.

### Внутренние ошибки

Внутренние ошибки могут возникать в двух случаях:

1) Введены некорректные данные. На этот случай предусмотреть защиту: при вводе данных некорректного формата необходимо оповестить пользователя об ошибке и попросить ввести данные еще раз.

2) Попытка удаления несуществующей записи из списка.

Данные ошибки должны обрабатываться программой, при этом пользователю должно быть сообщено об ошибке и предоставлена возможность продолжить работу.

3) Иные непредвиденные ошибки, которые будут обработаны в случае обнаружения во время тестирования.

## Интерфейс

### Основные требования

*Интерфейс должен быть реализован в консольном режиме.*

*Пользовательское меню должно содержать следующие пункты:*

1. Чтение данных из файла
2. Просмотр списков
3. Список групп сладостей
4. Список всех сладостей
5. Список сформированных подарков
6. Список всех сладостей в сформированных подарках
7. Назад
8. Сортировка
9. Поиск (сладостей)
10. По коду
11. По названию
12. По верхней границе цены
13. По верхней границе процентного содержания сахара.
14. Назад
15. Добавление
16. Добавление группы
17. Добавление сладости
18. Назад
19. Удаление
20. Удаление группы (вместе со всеми сладостями)
21. Удаление сладости
22. Назад
23. Редактирование полей сладостей
24. Формирование подарка
25. Выход из программы без сохранения изменений

10. Выход из программы с сохранением изменений

Как можно видеть, некоторые пункты меню содержат в себе подпункты для уточнения своей работы.

Представление форм ввода данных

* При выборе функции поиска будет возможность ввести следующие поля: код сладости, наименование, верхняя граница цены, верхняя граница процентного содержания сахара.
* При изменении записи пользователю будет предоставлена возможность выбора поля для изменения. Чтобы выйти из процесса редактирования на любом этапе необходимо ввести “q” или пустую строку.
* При удалении пользователь должен ввести название или id позиции к удалению (если удаляется группа, то удаляются и все сладости внутри). Чтобы выйти из процесса удаления необходимо ввести “q” или пустую строку.
* При добавлении пользователю предлагается вводить поочередно: название, цену, вес, сахар. Чтобы выйти на любом этапе ввода и не добавлять позицию в список необходимо ввести букву “q” или пустую строку.

## Требования к видам обеспечения

### Требования к аппаратному обеспечению

Процессор: 800 МГц

Оперативная память: 256 МБ оперативной памяти.

Память: 256 МБ свободной памяти на диске.

Клавиатура.

### Требования к программному обеспечению

Операционная система Windows 10 и выше.

### Требования к лингвистическому обеспечению

Поддержка русского языка в рабочей системе.

#### Требования к языкам и средствам манипулирования данными

Данное программное средство должно поддерживать ввод данных на русском и английском языках. Интерфейс программы должен быть на русском языке.

***СОСТАВИЛИ***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Наименование организации, предприятия*** | ***Должность исполнителя*** | ***Фамилия, имя, отчество*** | ***Подпись*** | ***Дата*** |
|  |  |  |  |  |

***СОГЛАСОВАНО***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Наименование организации, предприятия*** | ***Должность исполнителя*** | ***Фамилия, имя, отчество*** | ***Подпись*** | ***Дата*** |
|  |  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |  |