МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ

РЕСПУБЛИКИ КАЗАХСТАН

КОКШЕТАУСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

им. Ш.УАЛИХАНОВА

Физико-математический факультет

Кафедра «Информационных систем и вычислительной техники»

КУРСОВОЙ ПРОЕКТ

По дисциплине:

Технология программирования

на тему: «База данных Магазин товаров»

Выполнил: *студент 1 курса*

*группы «Вычислительная техника и*

*программное обеспечение »*

*Горчинский А.с.*

Проверил: *ст. преподаватель*

*Баклхазова У.У.*

г. Кокшетау 2010

# Содержание

Введение

1. Технология программирования на С++

1.1 Технология программирования

1.2 Основные этапы ее развития

2. База данных

1.1 Понятие базы данных

2.2 Характеристика баз данных

2.3 Основные типы баз данных

3. Действие программы

3.1 Структура программы

3.2 Работа программы

Заключение

Список литературы

Приложение

Введение

Базы данных всегда были важнейшей темой при изучении информационных систем. Однако в последние годы всплеск популярности Интернета и бурное развитие новых технологий для Интернета сделали знание технологии баз данных для многих одним из актуальнейших путей карьеры. Технологии баз данных увели Интернет-приложения далеко от простых брошюрных публикаций, которые характеризовали ранние приложения. В то же время Интернет-технология обеспечивает пользователям стандартизированные и доступные средства публикации содержимого баз данных. Правда, ни одна из этих новых разработок не отменяет необходимости в классических приложениях баз данных, которые появились еще до развития Интернета для нужд бизнеса. Это только расширяет важность знания баз данных.

Цель базы данных — помочь людям и организациям вести учет определенных вещей. На первый взгляд, эта цель кажется скромной, и вы, возможно, удивитесь, зачем нам нужна такая сложная технология и целый курс, посвященный этому предмету. Большинство из нас может вспомнить ситуации, в которых нам требуется отслеживать некоторые вещи. На самых ранних стадиях развития информационных технологий использовались списки — набитые на перфокарте и написанные на магнитной ленте. Со временем, однако, стало ясно, что только немногие проблемы можно решить с помощью таких списков. В следующем разделе мы обсудим такие проблемы, а затем опишем, как построить базы данных для их решения.

Процесс подготовки и решения задач на компьютерах состоит из нескольких этапов:

- постановка задачи;

- построение модели;

- разработка алгоритма;

- написание и отладка программы на языке программирования;

- тестирование программы.

При разработке программ применяют метод структурного программирования: разделение задачи на относительно независимые части и выполнение отладки этих частей. При этом достигается качество и быстрота разработки программ.

Развитие вычислительной техники с самого начала было направлено на облегчение и ускорение ввода, поиска данных. Базы данных используются во многих предприятиях, отраслях науки и т.д. Разработано множество готовых языков программирования с помощью которых можно создавать базы данных.

1. Технология программирования на С++

* 1. Технология программирования

Программирование — сравнительно молодая и быстро развивающаяся отрасль науки и техники. Опыт ведения реальных разработок и совершенствования имеющихся программных и технических средств постоянно переосмысливается, в результате чего появляются новые методы, методологии и технологии, которые, в свою очередь, служат основой более современных средств разработки программного обеспечения.

*Технологией программирования* называют совокупность методов и средств, используемых в процессе разработки программного обеспечения. Как любая другая технология, технология программирования представляет собой набор технологических инструкций, включающих:

указание последовательности выполнения технологических операций;

перечисление условий, при которых выполняется та или иная операция;

описания самих операций, где для каждой операции определены исходные данные, результаты, а также инструкции, нормативы, стандарты, критерии и методы оценки и т. п. (рис. 1.1).

Кроме набора операций и их последовательности, технология также определяет способ описания проектируемой системы, точнее модели, используемой на конкретном этапе разработки.

Структура описания технологической операции определяющий совокупность методов, используемых на разных этапах разработки, или *методологию*.

Чтобы разобраться в существующих технологиях программирования и определить основные тенденции их развития, целесообразно рассматривать эти технологии в историческом контексте, выделяя основные этапы развития программирования, как науки.

1.2 Основные этапы ее развития

Первый этап - «стихийное» программирование. Этот этап охватывает период от момента появления первых вычислительных машин до середины 60-х годов XX в. В этот период практически отсутствовали сформулированные технологии, и программирование фактически было искусством. Первые программы имели простейшую структуру. Они состояли из собственно программы на машинном языке и обрабатываемых ею данных (рис. 1.2). Сложность программ в машинных кодах ограничивалась способностью программиста одновременно мысленно отслеживать последовательность выполняемых операций и местонахождение данных при программировании.

Появление ассемблеров позволило вместо двоичных или 16-ричных кодов использовать символические имена данных и мнемоники кодов операций. В результате программы стали более «читаемыми».

Создание языков программирования высокого уровня, таких, как FORTRAN и ALGOL, существенно упростило программирование вычислений, снизив уровень детализации операций.

Это, в свою очередь, позволило увеличить сложность программ.

Идея написания подпрограмм появилась гораздо раньше, но отсутствие средств поддержки в первых языковых средствах существенно снижало эффективность их применения.) Подпрограммы можно было сохранять и использовать в других программах. В результате были созданы огромные библиотеки расчетных и служебных подпрограмм, которые по мере надобности вызывались из разрабатываемой программы.

Типичная программа того времени состояла из основной программы, области *глобальных данных* и набора подпрограмм (в основном библиотечных), выполняющих обработку всех данных или их части.

Слабым местом такой архитектуры было то, что при увеличении количества подпрограмм возрастала вероятность искажения части глобальных данных какой-либо подпрограммой. Например, подпрограмма поиска корней уравнения на заданном интервале по методу деления отрезка пополам меняет величину интервала. Если при выходе из подпрограммы не предусмотреть восстановления первоначального интервала, то в глобальной области окажется неверное значение интервала. Чтобы сократить количество таких ошибок, было предложено в подпрограммах размещать *локальные данные*

Сложность разрабатываемого программного обеспечения при использовании подпрограмм с локальными данными по-прежнему ограничивалась возможностью программиста отслеживать процессы обработки данных, но уже на новом уровне. Однако появление средств поддержки подпрограмм позволило осуществлять разработку программного обеспечения нескольким программистам параллельно.

В начале 60-х годов XX в. разразился «кризис программирования». Он выражался в том, что фирмы, взявшиеся за разработку сложного программного обеспечения, такого, как операционные системы, срывали все сроки завершения проектов. Проект устаревал раньше, чем был готов к внедрению, увеличивалась его стоимость, и в результате многие проекты так никогда и не были завершены.

Объективно все это было вызвано несовершенством технологии программирования. Прежде всего стихийно использовалась разработка «снизу-вверх» - подход, при котором вначале проектировали и реализовывали сравнительно простые подпрограммы, из которых затем пытались построить сложную программу. В отсутствии четких моделей описания подпрограмм и методов их проектирования создание каждой подпрограммы превращалось в непростую задачу, интерфейсы подпрограмм получались сложными, и при сборке программного продукта выявлялось большое количество ошибок согласования. Исправление таких ошибок, как правило, требовало серьезного изменения уже разработанных подпрограмм, что еще более осложняло ситу­ацию, так как при этом в программу часто вносились новые ошибки, которые также необходимо было исправлять... В конечном итоге процесс тестирования и отладки программ занимал более 80 % времени разработки, если вообще когда-нибудь заканчивался. На повестке дня самым серьезным образом стоял вопрос разработки технологии создания сложных программных про­дуктов, снижающей вероятность ошибок проектирования.

Анализ причин возникновения большинства ошибок позволил сформулировать новый подход к программированию, который был назван «структурным» .

Второй этап - структурный подход к программированию (60-70-е годы XX в.).

*Структурный подход к программированию* представляет собой совокупность рекомендуемых

технологических приемов, охватывающих выполнение всех этапов разработки программного обеспечения. В основе структурного подхода лежит *декомпозиция* (разбиение на части) сложных систем с целью последующей реализации в виде отдельных небольших (до 40 - 50 операторов) подпрограмм. С появлением других принципов декомпозиции (объектного, логического и т. д.) данный способ получил название *процедурной* декомпозиции.

В отличие от используемого ранее процедурного подхода к декомпозиции, структурный подход требовал представления задачи в виде иерархии подзадач простейшей структуры. Проектирование, таким образом, осуществлялось «сверху вниз» и подразумевало реализацию общей идеи, обеспечивая проработку интерфейсов подпрограмм. Одновременно вводились ог­раничения на конструкции алгоритмов, рекомендовались формальные модели их описания, а также специальный метод проектирования алгоритмов - метод пошаговой детализации.

Поддержка принципов структурного программирования была заложена в основу так называемых *процедурных* языков программирования. Как правило, они включали основные «структурные» операторы передачи управления, поддерживали вложение подпрограмм, локализацию и ограничение области «видимости» данных. Среди наиболее известных языков этой группы стоит назвать PL/1, ALGOL-68, Pascal, С.

Одновременно со структурным программированием появилось огромное количество языков, базирующихся на других концепциях, но большинство из них не выдержало конкуренции. Какие-то языки были просто забыты, идеи других были в дальнейшем использованы в следующих версиях развиваемых языков.

Дальнейший рост сложности и размеров разрабатываемого программного обеспечения потребовал развития *структурирования данных*. Как следствие этого в языках появляется возможность определения пользовательских типов данных. Одновременно усилилось стремление разграничить доступ к глобальным данным программы, чтобы уменьшить количество ошибок, возникающих при работе с глобальными данными. В результате появилась и начала развиваться технология модульного программирования.

*Модульное программирование* предполагает выделение групп подпрограмм, использующих одни и те же глобальные данные в отдельно компилируемые *модули* (библиотеки подпрограмм), например, модуль графических ресурсов, модуль подпрограмм вывода на принтер (рис. 1.5). Связи между модулями при использовании данной технологии осуществляются через специальный интерфейс, в то время как доступ к реализации модуля (телам подпрограмм и некоторым «внутренним» переменным) запрещен. Эту технологию поддерживают современные версии языков Pascal и С (C++), языки Ада и Modula.

Использование модульного программирования существенно упростило разработку программного обеспечения несколькими программистами. Теперь каждый из них мог разрабатывать свои модули независимо, обеспечивая взаимодействие модулей через специально оговоренные межмодульные интерфейсы. Кроме того, модули в дальнейшем без изменений можно было использовать в других разработках, что повысило производительность труда программистов.

Практика показала, что структурный подход в сочетании с модульным программированием позволяет получать достаточно надежные программы, размер которых *не превышает 100 000 операторов*. Узким местом модульного программирования является то, что ошибка в интерфейсе при вызове подпрограммы выявляется только при выполнении программы (из-за раздельной компиляции модулей обнаружить эти ошибки раньше невозможно). При увеличении размера программы обычно возрастает сложность межмодульных интерфейсов, и с некоторого момента предусмотреть взаимовлияние отдельных частей программы становится практически невозможно. Для разработки программного обеспечения большого объема было предложено использовать *объектный подход.*

Третий этап - объектный подход к программированию (с середины 80-х до конца 90-х годов XX в.). *Объектно-ориентированное программирование* определяется как технология создания сложного программного обеспечения, основанная на представлении программы в виде совокупности *объектов,* каждый из которых является экземпляром определенного типа *{класса),* а классы образуют иерархию с *наследованием* свойств. Взаимодействие программных объектов в такой системе осуществляется путем передачи *сообщений*.

Основным достоинством объектно-ориентированного программирования по сравнению с модульным программированием является «более естественная» декомпозиция программного обеспечения, которая существенно облегчает его разработку. Это приводит к более полной локализации данных и интегрированию их с подпрограммами обработки, что позволяет вести практически независимую разработку отдельных частей (объектов) программы. Кроме этого, объектный подход предлагает новые способы организации программ, основанные на механизмах наследования, полиморфизма, композиции, наполнения. Эти механизмы позволяют конструировать сложные объекты из сравнительно простых. В результате существенно увеличивается показатель повторного использования кодов и появляется возможность создания библиотек классов для различных применений.

Бурное развитие технологий программирования, основанных на объектном подходе, позволило решить многие проблемы. Так были созданы среды, поддерживающие *визуальное программирование,* например, Delphi, C++ Builder, Visual C++ и т. д. При использовании визуальной среды у программиста появляется возможность проектировать некоторую часть, например, интерфейсы будущего продукта, с применением визуальных средств добавления и настройки специальных библиотечных компонентов. Результатом визуального проектирования является заготовка будущей программы, в которую уже внесены соответствующие коды.

2. База данных

2.1 Понятие База данных

База данных — организованная в соответствии с определёнными правилами и поддерживаемая в памяти компьютера совокупность данных, характеризующая актуальное состояние некоторой предметной области и используемая для удовлетворения информационных потребностей пользователей.

База данных — совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных.

База данных — совокупность взаимосвязанных данных, совместно хранимых в одном или нескольких компьютерных файлах.

Существует множество других определений, отражающих скорее субъективное мнение тех или иных авторов о том, что означает этот термин в их понимании, однако общепризнанная единая формулировка отсутствует. Наиболее часто используются следующие отличительные признаки: База данных хранится и обрабатывается в вычислительной системе. Таким образом, любые внекомпьютерные хранилища информации (архивы, библиотеки, картотеки и т. п.) базами данных не являются.

Данные в базе данных логически структурированы (систематизированы) с целью обеспечения возможности их эффективного поиска и обработки в вычислительной системе.

Структурированность подразумевает явное выделение составных частей (элементов), связей между ними, а также типизацию элементов и связей, при которой с типом элемента (связи) соотносится определённая семантика и допустимые операции.

Из перечисленных признаков только первый является строгим, а другие допускает различные трактовки и различные степени оценки. Можно лишь установить некоторую степень соответствия требованиям к БД.

Многие специалисты указывают на распространённую ошибку, состоящую в некорректном использовании термина база данных вместо термина система управления базами данных. Эти понятия, следовательно, необходимо различать.

Базы данных необходимы для хранения и фильтрации информации, для её обработки и обмена. В целом, спектр применения достаточно широкий. По всему миру люди используют БД, начиная от простого пользователя-любителя, заканчивая крупнейшими мировыми корпорациями.

База данных – это хранилище для большого количества систематизированных данных, с которыми можно производить определённые действия. Под действиями понимается добавление, удаление, изменение, копирование, упорядочивание и прочее.

Все данные в БД можно представить в виде записей или объектов.

Для успешной работы с БД необходимы какие-то программные средства, при помощи которых можно было бы создавать и управлять БД. Для этого существуют Системы Управления Базами Данных (СУБД).

СУБД – совокупность языковых и программных средств, обеспечивающих создание, использование и ведение БД.

Существует два вида СУБД: локальные и сетевые.

Локальные – это СУБД, работающие на одном компьютере. К таким относятся dBase, FoxPro, Microsoft Access, Paradox и т.д.

Сетевые – это СУБД, позволяющие нескольким компьютерам использовать одну и ту же БД с помощью технологии клиент-сервер. Примером таких СУБД являются InterBase, Oracle, Microsoft SQL Server и т.д.Поскольку мы разбираем общие понятия, то расскажу немного о взаимосвязи данных.

Существует 4 типа взаимосвязи данных:

1) Один к одному

2) Один ко многим

3) Много к одному

4) Много ко многим

Вид взаимосвязи один к одному означает, что каждая запись одного объекта БД будет указывать на единственную запись другого объекта.

Один ко многим означает, что одной записи объекта БД будет соответствовать несколько записей других объектов.

Много к одному означает, что нескольким записям объектов БД будет соответствовать одна запись другого объекта.

Много ко многим устанавливается между двумя типами объектов БД.

2.2 Характеристика баз данных

База данных - это реализованная с помощью компьютера информационная структура (модель), отражающая состояние объектов и их отношения.

Следует учесть, что это определение не является единственно возможным. Информатика в отношении определений чаще всего не похожа на математику с ее полной однозначностью. Если подойти к понятию “база данных” с чисто пользовательской точки зрения, то возникает другое определение: база данных - совокупность хранимых операционных данных некоторого предприятия. Все дело в том, какой аспект доминирует в рассмотрении; в данной главе первое из определений более уместно.

Опыт использования баз данных позволяет выделить общий набор их рабочих характеристик:

• полнота - чем полнее база данных, тем вероятнее, что она содержит нужную информацию (однако, не должно быть избыточной информации);

• правильная организация - чем лучше структурирована база данных, тем легче в ней найти необходимые сведения;

• актуальность - любая база данных может быть точной и полной, если она постоянно обновляется, т.е. необходимо, чтобы база данных в каждый момент времени полностью соответствовала состоянию отображаемого ею объекта;

• удобство для использования - база данных должна быть проста и удобна в использовании и иметь развитые методы доступа к любой части информации.

Соответственно возможностям организации реляционных, иерархических и сетевых информационых структур, существуют и аналогичные виды баз данных.

2.3 Основные типы баз данных

1) Иерархические

2) Сетевые

3) Реляционные

Иерархические базы данных

Иерархические БД применялись в начале 60-х годов. Они построены в виде обычного дерева. Данные делятся на 2 категории: главные и подчинённые. Таким образом, один тип объекта является главным, а остальные, находящиеся на более низких ступенях иерархии, - подчинёнными.БД, организованные по такому принципу, удобно использовать в тех случаячх, когда информамция упорядочена соответствующим образом.

Сетевые базы данных

Сетевые БД начали применятся практически одновременно с иерархическими. В этих БД любой объект может быть как главным, так и подчинённым.

В использовании довольно сложно реализовать представление данных в таком виде, поэтому от этого типа тоже отказались.

Реляционные базы данных

Именно реляционные БД используются в повседневной жизни. (от англ. relation - отношение). Такой тип БД представляется собой несколько взаимосвязанных таблиц прямоугольной формы. В которых собственное есть поля, ключи, записи, аторибуты и прочее и прочее.

3. Действие программы

* 1. Структура программы

Программа состоит из 2 функций отвечающих за интерфейс программы и основной программы в которой выполняются все действия с базой данных.

Функции отвечающие за интерфейс программы предназначены для придания программе более приятного вида и для упрощения работы с ней. Это функция вывода сообщения об ошибке ввода и функция создания рамки в окне программы.

Функция вывода сообщения об ошибке ввода предназначена для упрощения работы с программой. Вообще сама «защита» от ошибки ввода защищает программу от совершения каких-либо ошибок, связанных с вводом. Если пользователь случайно введет неправильное значение то программа выдаст ему соответствующее сообщение и пользователь сможет исправить свою ошибку.

Функция создания рамки в окне программы предназначена для придания интерфейсу программы более «товарного» вида. Пользователю будет приятнее работать в красивом окне чем в простом черном окне.

Основная программа выполняет действия над записями: ввод - вывод данных, изменение записей, поиск записей, добавление записей к уже имеющимся.

3.2 Работа программы

Программа начинается с ввода первоначальных данных.

При запуске появляется окно в котором пользователю предлагается ввести нужные данные. Окно выглядит так:

После ввода данных пользователю предлагается продолжить ввод данных или же завершить ввод.

В случае неправильного ввода программа выдает сообщение об ошибке:

Если пользователь вводит символ 'n' то программа переходит к окну ввода данных, Если же пользователь ввел символ 'q' то в окне появляется сообщение о завершении ввода.

Программа переходит к следующему окну

Далее пользователю предлагается выбрать дальнейшее действие.

При выборе вывода данных на экране появляется окно с введенными данными

При выборе добавления данных на экране появляется окно ввода данных. При выборе поиска на экране появляется окно выбора критерия поиска:

Далее появляется окно в котором предлагается ввести определенные данные для поиска. После выполнения поиска при наличии данной записи она выводится на экран.

При выборе изменения данных на экране появится окно в котором пользователю предлагается ввести данные записи которую нужно изменить

Далее пользователю предлагается ввести новые данные.

Далее выводится сообщение об изменении записи.

При выборе выхода из программы программа завершает свое выполнение и заврывается.

# Заключение

Данная работа выполнена на языке высокого уровня программирования в среде Borland С++, Первые версии языка программирования С++ (тогда он назывался "Си с классами") были разработаны в начале 80-х годов Бьярном Страуструпом, сотрудником знаменитой AT&T Bell Labs, где ранее были разработаны такие шедевры программирования, как операционная система UNIX и язык программирования Си. При написании и разработки программного обеспечения использовались различные возможности С++.

В ходе выполнения курсовой работы была разработана база данных группы студентов, с помощью которой можно добавить данные о студенте, просмотреть их , совершить поиск по определенным критериям студента.

Также данные записываются в файл на жесткий диск, тем самым мы сохраняем их. Уже поиск совершается в самом файле.

Данная курсовая работа позволила укрепить знания, которые были получены в процессе учебы, и реализовать их в виде данной программы.

Список литературы

1. Е Зуев, А Кротов "Новые возможности языка Си++", PC Magazine/Russian Edition, #7, 1994.

2. В.Шелест. Программирование. 2002.

3. Г. Шилдт «Самоучитель С++» - СПб.:БХВ-Петербург. 2001. — 670с.

4. Карпов Б., Баранов Т. «С++: специальный справочник» - СПб.: Питер, 2001. — 480 с.:ил.

5. Культин Н. «С/С++ в задачах и примерах» - СПб.:БХВ-Петербург, 2002. — 288 с.

6. А. Мешков, Ю. Тихомиров «Visual C++ и MFC» - СПб.:БХВ-Петербург. 2002 — 1017с.

# 7. Теллес, М. BORLAND C++ BUILDER: библиотека программиста .-- СПб : Питер, 1998 .-- 512с.

# 8. Архангельский, А.Я. Язык C++ в C++Builder : справочное и методическое пособие / А.Я. Архангельский .-- М. : Бином, 2008 .-- 942 с.

9. Роберт Седжвик, Фундаментальные алгоритмы на С++,

Санкт-Петербург, ООО "ДиаСофтЮП", 2002. Части 1-4 - 688 с., часть 5 - 496 с.

10. М. Эллис, Б. Строуструп. Справочное руководство по языку C++ с комментариями: Пер. с англ. - Москва: Мир, 1992. 445с.

11. Стенли Б. Липпман. C++ для начинающих: Пер. с англ. 2тт. - Москва: Унитех; Рязань: Гэлион, 1992, 304-345сс.

12. Бруно Бабэ. Просто и ясно о Borland C++: Пер. с англ. - Москва: БИНОМ, 1994. 400с.

13. В.В. Подбельский. Язык C++: Учебное пособие. - Москва: Финансы и статистика, 1995. 560с.

14. Ирэ Пол. Объектно-ориентированное программирование с использованием C++: Пер. с англ. - Киев: НИИПФ ДиаСофт Лтд, 1995. 480с.

15. Т. Фейсон. Объектно-ориентированное программирование на Borland C++ 4.5: Пер. с англ. - Киев: Диалектика, 1996. 544с.

Приложение

Листинг программы

#include <clx.h>

#include <stdlib.h>

#include <iostream.h>

#include <conio.h>

#include <stdio.h>

#include <math.h>

void ramka(),error();

void error()

{textcolor(4);

cprintf("Nepravil'nyi vybor");

textcolor(15);}

void ramka ()

{int xx1,yy1,xx2,yy2;

int xi;

xx1=2; yy1=1; xx2=78; yy2=24;

for (xi=xx1;xi<=xx2-1;xi++)

{textcolor(14);

gotoxy(xi,yy1);

cprintf("Н");

gotoxy(xi,yy2);

cprintf("Н");}

for (xi=yy1;xi<=yy2;xi++)

{gotoxy(xx1,xi);

cprintf("є");

gotoxy(xx2,xi);

cprintf("є");}

gotoxy(xx1,yy1);

cprintf("Й");

gotoxy(xx2,yy1);

cprintf("»");

gotoxy(xx1,yy2);

cprintf("И");

gotoxy(xx2,yy2);

cprintf("ј");

textcolor(15);}

void main()

{const int n=9999;

int m,i,j;

char l,k,x,y;

char vybor[25],vtip[25],vproizv[25],vmodel[25];

struct detal {int nomer; char naim[25]; char proizv[25]; char model[25];};

detal baza[n];

textmode(C80);

textbackground(1);

clrscr();

ramka();

gotoxy(25,5);

cprintf ("Vvedite dannye\n");

i=0;

metka2:

j=i;

gotoxy(25,7);

cprintf ("Tip produkta :");

cscanf ("%s",&baza[i].naim);

gotoxy(25,8);

cprintf ("Proizvoditel' :");

cscanf ("%s",&baza[i].proizv);

gotoxy(25,9);

cprintf ("Model' produkta :");

cscanf ("%s",&baza[i].model);

i++;

metka4:clrscr();

ramka();

gotoxy(8,9);

cprintf("Nazhmite 'n' dlya dobavleniya zapisi ili 'q' dlya zaversheniya vvoda\n");

gotoxy(33,10);

cscanf("%s",&x);

if (x=='n' || x=='q')

{if (x=='n')

{clrscr();

goto metka2;}

else {gotoxy(27,11);

cprintf ("zapisi dobavleny\n");}}

else

{gotoxy (27,11);

error();

getch(); getch();

goto metka4;}

getch(); getch();

metka1:clrscr();

ramka();

gotoxy(25,5);

cprintf("Vyberite deistvie");

gotoxy(25,7);

cprintf("1-Vyvesti spisok na ekran");

gotoxy(25,8);

cprintf("2-Dobavit' dannye v spisok");

gotoxy(25,9);

cprintf("3-Vypolnit' poisk");

gotoxy(25,10);

cprintf("4-Izmenit' zapis'");

gotoxy(25,11);

cprintf("0-vyhod");

gotoxy(25,13);

cprintf("Vash vybor : ");

cscanf("%s",&k);

if (k=='0' || k=='1' || k=='2' || k=='3' || k=='4')

{if (k=='1')

{clrscr();

ramka();

for (m=0;m<i;m++)

{gotoxy(20,m+2);

cprintf (baza[m].naim);

cprintf(" ");

cprintf (baza[m].proizv);

cprintf(" ");

cprintf (baza[m].model);

printf("\n");}

getch(); getch();

goto metka1;}

if (k=='2')

{clrscr();

gotoxy(25,5);

cprintf ("Vvedite dannye\n");

metka3:j=i;

ramka();

gotoxy(25,7);

cprintf ("Tip produkta :");

cscanf ("%s",&baza[i].naim);

gotoxy(25,8);

cprintf ("Proizvoditel' :");

cscanf ("%s",&baza[i].proizv);

gotoxy(25,9);

cprintf ("Model' produkta :");

cscanf ("%s",&baza[i].model);

i++;

metka6:clrscr();

ramka();

gotoxy(9,8);

cprintf("Nazhmite 'n' dlya dobavleniya zapisi ili 'q' dlya zaversheniya vvoda\n");

gotoxy(30,9);

cscanf("%s",&y);

if (y=='n' || y=='q')

{if (y=='n') {clrscr();

goto metka3;}

else {gotoxy(27,11);

cprintf ("Zapisi dobavleny");

getch(); getch();

goto metka1;}}

else

{gotoxy (27,11);

error();

getch(); getch();

goto metka6;}}

if (k=='3')

{metka5:clrscr();

ramka();

gotoxy(25,5);

cprintf("Kryterii poiska");

gotoxy(25,7);

cprintf("1-Tip");

gotoxy(25,8);

cprintf("2-Proizvoditel'");

gotoxy(25,9);

cprintf("3-model' produkta");

gotoxy(25,11);

cprintf("Vash vybor : ");

cscanf("%s",&l);

if (l=='1' || l=='2' || l=='3')

{if (l=='1')

{clrscr();

ramka();

gotoxy(25,5);

cprintf("Vvedite tip produkta");

gotoxy(30,7);

cscanf("%s",&vybor);

for (m=0;m<i;m++)

{if (!strcmp(baza[m].naim,vybor))

{gotoxy(20,m+9);

cprintf (baza[m].naim);

cprintf(" ");

cprintf (baza[m].proizv);

cprintf(" ");

cprintf (baza[m].model);

cprintf("\n");}}}

if (l=='2')

{clrscr();

ramka();

gotoxy(25,5);

cprintf("Vvedite proizvoditelya produkta");

gotoxy(30,7);

cscanf("%s",&vybor);

for (m=0;m<i;m++)

{if (!strcmp(baza[m].proizv,vybor))

{gotoxy(20,m+9);

cprintf (baza[m].naim);

cprintf(" ");

cprintf (baza[m].proizv);

cprintf(" ");

cprintf (baza[m].model);

cprintf("\n");}}}

if (l=='3')

{clrscr();

ramka();

gotoxy(25,5);

cprintf("Vvedite model' produkta");

gotoxy(30,7);

cscanf("%s",&vybor);

for (m=0;m<i;m++)

{if (!strcmp(baza[m].model,vybor))

{gotoxy(20,m+9);

cprintf (baza[m].naim);

cprintf(" ");

cprintf (baza[m].proizv);

cprintf(" ");

cprintf (baza[m].model);

cprintf("\n");}}}

getch(); getch();

goto metka1;}

else {gotoxy(27,12);

error();

getch(); getch();

goto metka5;}}

if(k=='4')

{clrscr();

ramka();

gotoxy (25,5);

cprintf("Vvedite zapis' kotoruyu hotite izmenit' :");

gotoxy (25,7);

cprintf("Tip - ");

cscanf("%s",&vtip);

gotoxy(25,8);

cprintf("Proizvoditel' - ");

cscanf("%s",&vproizv);

gotoxy(25,9);

cprintf("Model' - ");

cscanf("%s",&vmodel);

gotoxy(25,11);

cprintf("Vashi dannye prinyaty!");

getch(); getch();

for (m=0;m<i;m++)

{if (!strcmp(baza[m].naim,vtip) && !strcmp(baza[m].proizv,vproizv) && !strcmp(baza[m].model,vmodel))

{clrscr();

ramka();

gotoxy(25,5);

cprintf("Vvedite novye dannye: \n");

gotoxy(25,7);

cprintf("Tip - ");

cscanf("%s",&baza[m].naim);

gotoxy(25,8);

cprintf("Proizvoditel' - ");

cscanf("%s",&baza[m].proizv);

gotoxy(25,9);

cprintf("Model' - ");

cscanf("%s",&baza[m].model);

gotoxy(25,11);

cprintf("Zapis' izmenena");

getch(); getch();}}

goto metka1;}

if (k=='0') abort();}

else {gotoxy(27,12);

error();

getch(); getch();

goto metka1;}