1)

2)

3)

4) **Виды алгоритмических структур: циклические алгоритмы. Пример схемы алгоритма**

Цикл с предусловием начинается с проверки условия выхода из цикла. Это логическое выражение, например I<=6. Если оно истинно, то выполняются те действия, которые должны повторяться. В противном случае, если логическое выражение I<=6 ложно, то этот цикл прекращает свои действия.  
  
Цикл с постусловием функционирует иначе. Сначала выполняется один раз те действия, которые подлежат повторению, затем проверяется логическое выражение , определяющее условие выхода из цикла, например, I>6 .Проверка его осуществляется тоже по-другому. Если условие выхода истинно, то цикл с постусловием прекращает свою работу, в противном случае - происходит повторение действий, указанных в цикле. Повторяющиеся действия в цикле называются "телом цикла". Разновидности циклов приведены на рис. 10 а),б).

|  |  |
| --- | --- |
| http://techn.sstu.ru/kafedri/%D0%BF%D0%BE%D0%B4%D1%80%D0%B0%D0%B7%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F/1/MetMat/shaturn/prog/part7.files/ris1.gif | http://techn.sstu.ru/kafedri/%D0%BF%D0%BE%D0%B4%D1%80%D0%B0%D0%B7%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F/1/MetMat/shaturn/prog/part7.files/ris2.gif |
| a) Цикл с постусловием | б) Цикл с предусловием |
|  |  |

5) **Состав языка Си: алфавит, ликсемы, разделители.**

**Алфавит языка** — это тот набор символов (знаков), который допустим в данном языке. В алфавит языка C# входят:

* прописные и строчные буквы (латинские и национальных шрифтов) и символ подчеркивания (**\_**);
* цифры;
* специальные знаки: **' " , { } | [ ] ( ) + - / \ % ? ! . ; : < = > & \* ~ ^**
* неотображаемые символы («обобщенные пробельные символы»), они используются для отделения лексем друг от друга (это пробелы, табуляция, переход на новую строку).

Из отдельных символов алфавита языка строятся более крупные блоки программы: лексемы, директивы препроцессора и комментарии.

**Лексема** — это последовательность из одного или несколько символов, представляющая определенный смысл. Лексемам в языке человека соответствует понятие**слово**. В литературе, посвященной трансляции с языков программирования, часто используется термин **токен,** имеющий тот же смысл. Существует несколько видов лексем:

* идентификаторы (имена объектов);
* ключевые (зарезервированные, служебные) слова;
* знаки операций;
* разделители;
* литералы.

**Разделитель** используются для разделения или для группировки элементов. Примеры разделителей: пробелы, скобки, точка, запятая.

6) **Состав языка Си: ключевые слова, идентификаторы**

**Ключевые слова** — это служебные слова, которые зарезервированы в языке, их можно использовать только по прямому назначению (например, **for** — это заголовок оператора цикла и ничего более), т. е. зарезервированные слова нельзя использовать в качестве имен переменных пользователя.

**Идентификатор** — это последовательность букв, цифр и символов подчеркивания. Идентификаторы используются для обозначения имен объектов, используемых в программе. Всё, что применяется в программе, имеет имя. Имена (названия) имеют константы, переменные, методы, классы и т. д. Имя не может начинаться с цифры. Длина имени произвольная. Ограничения накладываются только на внешние имена, например, на имена библиотек, но это ограничения не языка C#, а операционной системы. В идентификаторах допускается применять национальные шрифты, например, русские буквы, но это нежелательно.

7)**Состав языка СИ: Константы и переменные.**

Все переменные до их использования должны быть определены (объявлены). При этом задается тип, а затем идет список из одной или более переменных этого типа, разделенных запятыми. Например: int a, b, c;

В языке различают понятия объявления переменной и ее определения. Объявление устанавливает свойства объекта: его тип (например, целый), размер (например, 4 байта) и т.д. Определение наряду с этим вызывает выделение памяти (в приведенном примере дано определение переменных).

Переменные в языке Си могут быть инициализированы при их определении: int a = 25, h = 6;

В языке возможны глобальные и локальные объекты. Первые определяются вне функций и, следовательно, доступны для любой из них. Локальные объекты по отношению к функциям являются внутренними. Они начинают существовать, при входе в функцию и уничтожаются после выхода из нее.

Наряду с переменными в языке существуют следующие виды констант:

* *вещественные,* например 123.456, 5.61е-4. Они могут снабжаться суффиксом F (или f), например 123.456F, 5.61e-4f;
* *целые,* например 125;
* *короткие целые,* в конце записи которых добавляется буква (суффикс) H (или h), например 275h, 344H;
* *длинные целые,* в конце записи которых добавляется буква (суффикс) L (или l), например 361327L;
* *беззнаковые,* в конце записи которых добавляется буква U (или u), например 62125U;
* *восьмеричные,* в которых перед первой значащей цифрой записывается нуль (0), например 071;
* *шестнадцатеричные,* в которых перед первой значащей цифрой записывается пара символов нуль-икс (0x), например 0x5F;
* *символьные* - единственный символ, заключенный в одинарные кавычки, например 'О', '2', '.' и т.п. Символы, не имеющие графического представления, можно записывать, используя специальные комбинации, например \n (код 10), \0 (код 0). Эти комбинации выглядят как два символа, хотя фактически это один символ. т.п.;
* *строковые* - последовательность из нуля символов и более, заключенная в двойные кавычки, например: "Это строковая константа". Кавычки не входят в строку, а лишь ограничивают ее. Строка представляет собой массив из перечисленных элементов, в конце которого помещается байт с символом '\0'. Таким образом, число байтов, необходимых для хранения строки, на единицу превышает число символов между двойными кавычками;
* *константное выражение,* состоящее из одних констант, которое вычисляется во время трансляции (например: а=60+301);
* типа *long double,* в конце записи которых добавляется буква L (или l), например: 1234567.89L.

8)**Состав языка СИ: Знаки операций (арифметические, инкремент и декремент, присваивания, отношения, логические)**

*Знак операции* — это один или более символов, определяющих действие над операндами. Внутри знака операции пробелы не допускаются. Например, в выражении a += b знак += является знаком операции, а a и b  — операндами. Символы, составляющие знак операций, могут быть специальными, например, +, &&, | и <, и буквенными, такими как as или new.

|  |  |
| --- | --- |
| Знак операции | Назначение операции |
| ( ) | Вызов функции |
| [ ] | Выделение элемента массива |
| **.** | Выделение элемента записи |
| -> | Выделение элемента записи |
| ! | Логическое отрицание |
| ~ | Поразрядное отрицание |
| - | Изменение знака |
| ++ | Увеличение на единицу |
| -- | Уменьшение на единицу |
| & | Взятие адреса |
| \* | Обращение по адресу |
| (тип) | Преобразование типа (т.е. (float) a) |
| sizeof( ) | Определение размера в байтах |
| \* | Умножение |
| / | Деление |
| % | Определение остатка от деления |
| + | Сложение |
| - | Вычитание |
| << | Сдвиг влево |
| >> | Сдвиг вправо |
| < | Меньше, чем |
| <= | Меньше или равно |
| > | Больше, чем |
| >= | Больше или равно |
| = = | Равно |
| != | Не равно |
| & | Поразрядное логическое "И" |
| ^ | Поразрядное исключающее "ИЛИ" |
| | | Поразрядное логическое "ИЛИ" |
| && | Логическое "И" |
| || | Логическое "ИЛИ" |
| ?: | Условная (тернарная) операция |
| = | Присваивание |
| +=, - =, \*=, /=, %=, <<=, >>=, &=, |=, ^= | Составные операции присваивания (например, а \*= b (т.е. a = a \* b) и т.д.) |
| **,** | Операция запятая |

9)

операция присваивания (=). Выражение вида

х = у;

присваивает переменной х значение переменной у. Операцию "=" разрешается использовать многократно в одном выражении, например:

x = y = z = 100;

Арифметические операции задаются следующими символами (табл. 2): +, -, \*, /, %. Последнюю из них нельзя применять к переменным вещественного типа. Например:

a = b + c;

x = y - z;

r = t \* v;

s = k / l;

p = q % w;

В следующей таблице перечислены приоритет и ассоциативность операторов C++. Операторы перечислены сверху вниз в порядке убывания приоритета.

|  |  |  |  |
| --- | --- | --- | --- |
| **Приоритет** | **Оператор** | **Описание** | **Ассоциативность** |
| **1** | :: | Область видимости | Слева-направо |
| **2** | ++   -- | Суффиксальный/постфиксный инкремент и декремент |
| () | Вызов функции |
| [] | Обращение к массиву по индексу |
| . | Выбор элемента по ссылке |
| −> | Выбор элемента по указателю |
| **3** | ++   -- | Префиксный инкремент и декремент | Справа-налево |
| +   − | Унарный плюс и минус |
| !   ~ | Логическое НЕ и побитовое НЕ |
| (*type*) | Приведение к типу type |
| \* | Indirection (разыменование) |
| & | Адрес |
| sizeof | Размер |
| new, new[] | Динамическое выделение памяти |
| delete,delete[] | Динамическое освобождение памяти |
| **4** | .\*   ->\* | Указатель на член | Слева-направо |
| **5** | \*   /   % | Умножение, деление и остаток |
| **6** | +   − | Сложение и вычитание |
| **7** | <<   >> | Побитовый сдвиг влево и сдвиг вправо |
| **8** | <   <= | Операторы сравнения < и ≤ |
| >   >= | Операторы сравнения > и ≥ |
| **9** | ==   != | Операторы сравнения = и ≠ |
| **10** | & | Побитовое И |
| **11** | ^ | Побитовый XOR (исключающее или) |
| **12** | | | Побитовое ИЛИ (inclusive or) |
| **13** | && | Логическое И |
| **14** | || | Логическое ИЛИ |
| **15** | ?: | Тернарное условие | Справа-налево |
| = | Прямое присваивание (предоставляемое по умолчанию для C++ классов) |
| +=   −= | Присвоение с суммированием и разностью |
| \*=   /=   %= | Присвоение с умножением, делением и остатком от деления |
| <<=   >>= | Присвоение с побитовым сдвигом слево и вправо |
| &=   ^=   |= | Присвоение с побитовыми логическими операциями (И, XOR, ИЛИ) |
| **16** | throw | Throw оператор (выброс исключений) |
| **17** | , | Запятая | Слева-направо |

10) Типы данных в языке Си: целый, вещественный, логический.

Тип данных определяет допустимый диапазон изменяемой переменной и постоянной, а так же допустимые операции на данные этого типа.

Каждая переменная и постоянная должны принадлежать определенному типу данных, т.к. выделен определенный объём памяти и проверяется правильность написания.

Над численными данными можно производить арифметические операции. А символьные данные используют для анализа текста.

**Целый тип данных**

Он используется для представления целых чисел. В его состав входят: char, int, short int, long int, unsigned int.

1. **Целый тип char**

Он определяется двумя способами:

1. Любое целое число от -128 до +127 и символы с исходными кодами.
2. Для типа char определенны операции отношения (=, > , < , ≤ , ≥ ). Результат выполнения –логического типа и стандартные функции.

Если данные типа char рассматривать как целые числа, то для них определены арифметические операции (+, - , × , / , % от числа). Результат выполнения этих операций имеет тип char, а операции отношения- результат логического типа и стандартные функции.

1. **Целый тип short int (короткая целая)**

Для данных этого типа определенны арифметические операции (+, - , × , / , % от числа). Результат выполнения- целый тип short int. Так же определены операции отношения (=, > , < , ≤ , ≥ ). Результат- логического типа.Определены и стандартные функции.

1. **Целый тип int**

Для данного типа определены арифметические операции (+, - , × , / , % от числа). Результат выполнения – целый тип int; операции отношения и стандартные функции.

1. **Целый тип long int**

Для данного типа определены арифметические операции (+, - , × , / , % от числа). Результат выполнения – целый тип long int; операции отношения и стандартные функции.

1. **Без знаковое целое (unsigned int)**

Для данного типа определены арифметические операции (+, - , × , / , % от числа). Результат выполнения – целый тип unsigned int; операции отношения и стандартные функции.

**Вещественные типы данных**

Используется дял задания чисел с не нулевой дробной частью.

1. **Тип float**

Для данного типа определены арифметические операции (+, - , × , / , % от числа). Результат выполнения – вещественный тип float. Определены операции отношения. Результат выполнения – логического типа.

1. **Тип double**

Для данного типа определены арифметические операции (+, - , × , / , % от числа). Результат выполнения – вещественный тип double. Определены операции отношения. Результат выполнения – логического типа.

1. **Тип long double**

Для данного типа определены арифметические операции (+, - , × , / , % от числа). Результат выполнения – вещественный тип double. Определены операции отношения. Результат выполнения – логического типа.

**Логический тип данных**

Представляет собой логическое значение: true, false.

При вычислении значений в арифметическом выражении значение = 0 если принимается за ложь (false), а в любое другое значение ≠0 является истиной (true).

11)

12)

13)

14)

15)

16)

17)

18)

19)

20)

21)

22)

23)**Одномерные массивы. Понятия и работа с одномерными массивами**

Массив - это непрерывный участок памяти, содержащий последовательность объектов одинакового типа, обозначаемый одним именем.

Массив характеризуется следующими основными понятиями:

1. Элемент массива – значение, хранящееся в определенной ячейке памяти, расположенной в пределах массива, а также адрес этой ячейки памяти.
2. Адрес массива – адрес начального элемента массива.
3. Имя массива – идентификатор, используемый для обращения к элементам массива.
4. Размер массива – количество элементов массива
5. Размер элемента – количество байт, занимаемых одним элементом массива

Стандартный вид объявления одномерного массива следующий:  
 тип имя\_переменной [размер];

В С массивы должны определяться однозначно (т.е иметь уникальное имя\_переменной), чтобы компилятор мог выделить для них место в памяти. Здесь тип объявляет базовый тип массива и является типом каждого элемента массива. Параметр размер определяет, сколько элементов содержит массив.

У всех массивов первый элемент имеет индекс 0. Поэтому, если написать  
char р [10];  
то будет объявлен массив символов из 10 элементов, причем эти элементы адресуются индексом от 0 до 9.

Пример программы иллюстрирующий работу с массивом

#include <stdio.h> //Подключаем библиотеку ввода-выводаint main(void) //определение главной функции{int x[10]; /\* резервирует место для 10 целочисленных элементов типа int \*/int t; /\*Создаём переменную для получения значения элемента массива \*/for(t=0; t<10; ++t) x[t] = t; /\* загружает целочисленный массив числами от 0 до 9 x[0]=0 x[1]=1 итд \*/for(t=0; t<10; ++t) printf("%d ", x[t]); /\*выводим значения элементов массива на экран начиная от x[0] до x[9] включительно\*/return 0;} **//end main**

24)

25)

26)

27)

28)

29)

30)

31) **Информационный объект. Понятие предметной области. Атрибуты данных. Понятие реквизита, понятие экземпляра.**

Информационный объект – описание некоторой сущности предметной области (объект). Информационный объект – образуется совокупностью логически взаимодействующих атрибутов, представляющие количественные и качественные характеристики объекта.

Экземпляр записи – отдельная реализация записи, содержащая конкретные значения её полей.

Реквизит — это неделимая с точки зрения смыслового описания предметной области единица информации, отражающая качественную или количественную характеристику объекта (процесса).

Предметная область – объект реального мира.

32**) Понятие ключа. Виды ключей. Назначение ключей.**

Ключ — это реквизит или группа реквизитов, которые используются для идентификации конкретной записи в реляционной таблице.

Первичный ключ – столбец или совокупность столбцов, значения которого однозначно идентифицируют запись в таблице.

Внешний ключ – столбец или совокупность столбцов, значения которого однозначно идентифицируют значения первичного ключа другого

33) **Понятие типа данных. Типы данных.**

Тип данных— множество значений и операций на этих значениях.

Целочисленные типы

Целочисленные типы содержат в себе значения, интерпретируемые как числа (знаковые и беззнаковые).

Строковые типы

Последовательность символов, которая рассматривается как единое целое в контексте переменной.

Абстрактные типы данных

Типы данных, которые рассматриваются независимо от контекста и реализации в конкретном языке программирования.

34) Понятие нормализации отношений. Первая нормальная форма (I НФ)  
Нормализация отношений.  
Нормализация отношений – это формальный аппарат ограничений на формирование отношений таблицы, которые позволяют устранить дублирование, обеспечение непротиворечивости данных, уменьшение затраты труда на ведение БД.

|  |  |
| --- | --- |
| Студенты |  |
| \*№ студ. билета |  |
| фамилия |  |
| имя | - I НФ |
| отчество |  |
| дата  рождения |  |

**Первая нормальная форма**Отношение находящиеся в I НФ, если все атрибуты просты и неделимы. **Пример:**Студенты (\*№ студ. билета, ФИО, дата рождения, телефон)Студенты (\*№ студ. билета, фамилия, имя , отчество дата рождения)

35) Понятие нормализации отношений. Вторая нормальная форма (II НФ)

Если отношения находятся в I НФ и каждый не ключевой атрибут функционально полно зависит от первичного ключа.

**Функциональная зависимость**реквизитов – зависимость, при которой в экземпляре информационного объекта определенному значению ключевого реквизита соответствует только одно значение описательного реквизита.

|  |  |
| --- | --- |
| Студенты |  |
| \*№ студ. билета |  |
| фамилия |  |
| имя | - II НФ |
| отчество |  |
| дата  рождения |  |

36) Понятие нормализации отношений. Третья нормальная форма (III НФ)  
Транзитивная зависимость- наблюдается в том случае если один из двух описываемых реквизитов зависит от первичного ключа, а другой описывает реквизит зависящий от 1-го описываемого реквизита.  
Отношения находятся в III НФ если находятся во II НФ и каждый не ключевой атрибут НЕ транзитивно зависим от первичного ключа.  
Для устранения транзитивной зависимости необходимо произвести расщепление исходного объекта.

37)

38)

39)

40)

41)

42)

43)

44)

45)

46) Определение СУБД. Назначение СУБД. Классификация СУБД.  
Система управления БД (СУБД) – это совокупность языковых и программных средств, предназначенных для создания, введения и совместного использования БД многими пользователями.  
Основная задача СУБД – предоставить пользователю БД возможность работать, не вникая в детали на уровне аппаратного обеспечения, в его распоряжение входит набор операций, выражаемых в терминах языка высокого уровня (например, набор операций, которые можно выполнять с помощью языка SQL).  
СУБД являются посредниками между логической структурой данных, необходимых разным приложениям, и физическими хранилищами данных (обычно это файловая система персонального компьютера или сервера, хотя последнее время хранилища могут распределяться между многими серверами). Физическая структура данных должна быть скрыта от программистов. СУБД должны хранить логическую структуру (метаданные), предотвращая несогласованные изменения данных, нарушающие эту структуру.  
Таким образом, любая СУБД должна обеспечивать следующее:  
1. компактное хранение данных (без дублирования);  
2. оптимизацию доступа к данным;  
3. логическую целостность (согласованность) данных;  
4. универсальный интерфейс (язык или протокол), позволяющий задавать структуру данных, изменять и извлекать их неизвестному заранее алгоритму.  
5. Обеспечение этих требований к информационным системам на уровне СУБД позволяет избегать повторения одной и той же работы при разработке программ.  
СУБД делятся на:  
СУБД общего назначения (СУБД ОН)   
СУБД специализированные  
СУБДОН – не ориентирован на какую-либо конкретную предметную область. Каждая система такого рода реализуется как программный продукт, способный функционировать на некоторых моделях ПК в определенной операционной обстановке.  
СУБДОН обладает средствами настройки на работу с конкретной БД в условиях конкретного применения.

47) Типовая организация СУБД.Логические современные реляционные СУБД можно разделить на следующие составляющие:  
1) Внутренняя часть – это ядро СУБД  
2) Компилятор языка БД (SQL язык)  
3) Подсистема поддержки времени выполнения  
4) Набор утилит  
Ядро СУБД отвечает за управление данными во внешней памяти; управление транзакциями; управление буферами оперативной памяти; журнализация.  
Выделяются следующие компоненты ядра: менеджер данных, менеджер буфера, менеджер транзакций и менеджер журнала.

48) Программные компоненты среды СУБД: процессор запросов, компилятор языка определения данных. Процессор запросов – преобразует запросы, последовательность низкоуровневых инструкций для контроля БД.  
Компилятор языка определения данных – преобразует команды набор таблиц, содержащих метаданные. Затем, эти таблицы сохраняются в системном каталоге, а управляющая информация сохраняется в заголовках данными.

49)

50)

51)

52)

53)

54)

55)

56)

57)

58)Объекты БД: таблицы. Способы создания таблиц. Режимы работы с таблицами.

**Таблица** – это объект БД предназначенный для хранения данных и полей.

Способы создания таблиц

Для создания таблиц используются следующие способы:

1. С помощью ввода дынных
2. С помощью шаблона
3. С помощью конструктора
4. **Создание таблиц с помощью ввода данных**

Это простой метод ввода данных

Недостатки: не все типы данных определяются системой, на этапе проектирования таблицы необходимо предусмотреть все типы связи.

1. **Создание с помощью шаблона**

Так же простой способ создания таблиц

Недостатки: малый выбор шаблонов, связи и типы данных неизменен.

1. **Создание таблиц с помощью конструктора**

В таблице осуществляется работа с данными; создаётся или модифицируется структура таблицы, т.е. задаются имена полей таблицы, их типы, определяются свойства.

59) Информационные блоки Конструктора таблиц

|  |  |  |
| --- | --- | --- |
| Имя поля | Тип данных | Описание |
|  |  |  |
|  |  |  |
| Свойства поля | | |
| Общие | Подстановка | |
|  |  | |
|  |  | |

В столбце «Имя поля» вводятся имена полей созданной таблицы; в столбце «тип данных» для каждого поля таблицы выбирается тип данных; в столбце «Описание» вводится описание данного поля.

В нижней части окна конструктора таблицы есть вкладка «Общие» и «Подстановка». Во вкладке «Общие» указы основные свойства поля.

60) Объекты БД: запросы. Типы запросов.

Запрос-это вопрос о данных, хранящихся в таблицах или инструкция на отбор записей, подлежащих изменению.

Запрос на выборку – задает запрос о данных, хранящихся в таблице.

Запрос изменения – изменение или перемещение данных.

К нему относятся:

1. Запрос на удаление(удаление группы записей из 1 или нескольких таблиц)
2. запрос на обновление (изменение значения полей таблицы)
3. запрос на добавление(добавление записей из 1 таблицы в другую)
4. запрос на создание таблицы(создание новой таблицы на основе всех данных или их части из 1 или нескольких таблиц)
5. Перекрестные запросы предназначены для группировки данных и представления их в компактном виде.

Запрос с параметрами- позволяет определить 1 или несколько условий отбора во время выполнения запроса.

Запросы SQL созд. с помощью инструкций SQL языка.

61) **Построение запросов в режиме Конструктора**

Данный способ используется для создания или изменения запросов. Окно конструктора запросов состоит из 2-х частей. В верхней части окна отображается таблица; таблицы, являющиеся источниками данных называют базовыми. В нижней части окна конструктора таблицы находится бланк запроса, ячейки которого используются для определения запроса.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Таб. 1  Таб. 2  Таб. 3 | | | | Базовая таблица |
| Параметры запроса | | | |  |
| Поле |  |  | Бланк запроса | |
| Имя таблицы |  |  |
| Вывод на экран |  |  |

62)**Построение запросов с использованием Мастера**

С его помощью создаются простые запросы, перекрестные запросы, повторяющиеся записи, записи без подчиненных.

63) **Перекрестные запросы**

Перекрестные запросы предназначены для группировки данных и предоставления их в компактном виде.

64) **Объекты БД: формы. Способы создания форм. Режимы работы с формами.**

Назначение форм

Формы в Microsoft Access создают для того, чтобы облегчить ввод и редактирование данных, обеспечить их вывод в удобном для пользователя представлении.

**Формы — это объекты, предназначенные, в основном, для ввода и отображения данных на экране, хотя они могут быть распечатаны и содержать так называемые элементы управления, такие как поля, списки, флажки, переключатели и др.**

В формы можно помещать командные кнопки для открытия других форм, выполнения запросов или команд меню, фильтрации выводимых на экран данных, организации вывода сообщений или печати информации (в частности, можно установить разные наборы опций для вывода формы на экран и на печать). Таким образом, формы позволяют управлять ходом выполнения приложения и являются основным средством организации интерфейса пользователя в Microsoft Access.

**Основные способы создания форм**

При создании формы следует указать имя таблицы или запрос, являющегося источником сведений и выбрать способ создания формы (помимо режима конструктора можно создать форму с помощью мастера, создать форму-диаграмму для повышения наглядности числовых данных или сводную таблицу типа сводной таблицы Microsoft Excel).

С помощью мастера форм можно создавать формы на основе одной или нескольких таблиц и/или запросов, а затем совершенствовать их в режиме конструктора. Мастер форм разбивает процесс создания формы на несколько этапов, на каждом из которых требуется установить определенные параметры для указания доступных полей, связей и внешнего вида формы.

**Вручную в режиме конструктора**. Сначала создается базовая форма, которая затем изменяется в соответствии с требованиями в режиме конструктора .

Автоформы в столбец, ленточная и табличная представляют собой разные варианты представления информации из исходной таблицы. Автоформа в столбец отображает каждую запись источника в виде набора элементов управления, соответствующих полям записи, расположенным в один столбец. В ленточной автоформе записи следуют друг за другом, и каждая отображается в виде набора элементов управления, соответствующих полям записи, расположенным в одну строку. Записи из источника данных табличной автоформеотображаются в виде таблицы.

Разработку структуры, изменение внешнего вида формы, добавление и удаление элементов управления можно производить вручную в режиме конструктора.

**Режимы работы с формами**

При работе с формами можно использовать три режима:

* режим конструктора форм для создания и редактирования формы,
* режим формы для ее просмотра,
* режим таблицы для просмотра источника данных формы.

65) **Объекты БД: формы. Структура формы.**

Структура формы такова: макет формы состоит из разделов. Любая форма может включать следующие разделы:

* раздел **Заголовок формы**(Form Header) определяет верхнюю часть формы. Этот раздел добавляется в форму вместе с разделом примечания формы. В область заголовка формы можно поместить текст, графику и другие элементы управления. При печати многостраничной формы раздел заголовка отображается только на первой странице;
* раздел **Верхний колонтитул**(Page Header) определяет верхний колонтитул страницы при печати формы. Этот раздел добавляется в форму вместе с разделом, определяющим нижний колонтитул страницы, и отображается только когда форма открыта в режиме Предварительного просмотра. При печати многостраничной формы верхний колонтитул отображается вверху каждой страницы;
* раздел **Область данных**(Detail) определяет основную часть формы, содержащую данные, полученные из источника. Данный раздел может содержать элементы управления, отображающие данные из таблиц и запросов, а также неизменяемые данные, например надписи. При печати многостраничной формы этот раздел отображается на каждой странице;
* раздел **Нижний колонтитул**(Page Footer) определяет нижний колонтитул страницы при печати формы. Этот раздел добавляется в форму вместе с разделом, определяющим верхний колонтитул страницы. Он отображается только когда форма открыта в режиме Предварительного просмотра. При печати многостраничной формы нижний колонтитул отображается внизу каждой страницы;
* раздел **Примечание формы**(Form Footer) определяет нижнюю часть формы. Этот раздел добавляется в форму вместе с разделом заголовка формы. При печати многостраничной формы примечание формы будет отображено только внизу последней страницы.

66) **Разновидности элементов управления по типу содержимого.**

В Access существует три разновидности элементов управления, в зависимости от типа содержимого в них, т. е. от способа заполнения их данными:

* присоединенные
* свободные
* вычисляемые

К какому из перечисленных видов относится тот или иной элемент управления, определяется тем, как задан для него источник данных.

*Присоединенные элементы управления*связаны с полями базовой таблицы, т. е. той таблицы, которая является источником данных для формы. Если источником данных является запрос, то присоединенные элементы управления могут связываться с полями в разных таблицах. В присоединенном элементе отображаются данные, которые содержатся в связанном с ним поле таблицы, и при изменении этих данных соответствующим образом обновляется и значение в поле таблицы. В присоединенных элементах можно отображать все доступные в Access типы данных, в том числе объекты OLE и гиперссылки.

*Свободные элементы управления*не связаны с таблицами. Они предназначены либо для ввода информации, которая используется не для непосредственного редактирования данных в источнике, а в других целях (обычно макросами или программами VBA), либо для отображения объектов OLE, которые хранятся в самих формах. Свободными элементами являются также все элементы, не связанные с какими-либо данными, а предназначенные лишь для улучшения визуального восприятия форм, такие как линии, прямоугольники, рисунки.

*Вычисляемые элементы управления —*это такие элементы, значения которых вычисляются на основе значений других элементов. В качестве источника данных для этих элементов используются выражения и функции.

67)

68)

69)

70)

71)

72)

73)

74)

75)

76) Оператор манипулирования данными, назначения и формы команд.

Операторы DML (Data Manipulation Language) - операторы манипулирования данными

* SELECT - отобрать строки из таблиц
* INSERT - добавить строки в таблицу
* UPDATE - изменить строки в таблице
* DELETE - удалить строки в таблице
* COMMIT - зафиксировать внесенные изменения
* ROLLBACK - откатить внесенные изменения

77) Оператор ввода данных INSERT. Формат команды и ее назначение.

Оператор ввода данных INSERT имеет следующий синтаксис:

INSERT INTO имя\_таблицы [(<список столбцов>) ] VALUES (<список значений>)  
Подобный синтаксис позволяет ввести только одну строку в таблицу. Задание списка столбцов необязательно тогда, когда мы вводим строку с заданием значений всех столбцов.   
78) Оператор удаления данных DELETE. Формат команды и ее назначение.  
Синтаксис оператора DELETE следующий:  
DELETE FROM имя\_таблицы [WHERE условия\_отбора]  
Если условия отбора не задаются, то из таблицы удаляются все строки, однако это не означает, что удаляется вся таблица. Исходная таблица остается, но она остается пустой, незаполненной.  
79) Оператор обновления данных UPDATE. Формат команды и ее назначение.  
Операция обновления данных UPDATE требуется тогда, когда происходят изменения во внешнем мире и их надо адекватно отразить в базе данных, так как надо всегда помнить, что база данных отражает некоторую предметную область.  
Синтаксис оператора UPDATE следующий  
UPDATE имя\_таблицы  
SET имя\_столбца = новое\_значение [WHERE условие\_отбора]

80) Операторы определения данных, назначения и формы команд.

Операторы DDL (Data Definition Language) - операторы определения объектов базы данных.

CREATE SCHEMA - создать схему базы данных

* DROP SHEMA - удалить схему базы данных
* CREATE TABLE - создать таблицу
* ALTER TABLE - изменить таблицу
* DROP TABLE - удалить таблицу
* CREATE DOMAIN - создать домен
* ALTER DOMAIN - изменить домен
* DROP DOMAIN - удалить домен
* CREATE COLLATION - создать последовательность
* DROP COLLATION - удалить последовательность
* CREATE VIEW - создать представление
* DROP VIEW - удалить представление

81)

82)