Вопрос 2. Написание программного кода с использованием языков программирования, определения и манипулирования данными.

1. Разработка программы

Процесс программирования – (в широком смысле) процесс создания программ; (в узком смысле) процесс кодирования на одном из языков программирования.

Система программирования — комплекс средств, предназначенных для создания и эксплуатации программ на конкретном языке программирования на ЭВМ определенного типа.

Примеры: *Turbo Pascal; Pascal ABC; Delphi; C++ Builder и др.*

Системы программирования включают в себя:

- текстовый редактор для написания кода;
- транслятор;
- набор библиотек различных данных;
- отладчик и др.
- дополнительные инструменты для автоматизации, тестирования и визуализации процесса разработки т.п.

Для обозначения таких систем часто используется термин **«интегрированная среда разработки»**, который означает, что предоставляется все необходимое для превращения кода в функционирующие приложения.

Примером *интегрированной среды разработки программного обеспечения, предназначенной для написания консольных приложений, приложений с графическим интерфейсом пользователя, веб-сайтов, веб-приложений, веб-служб и др., является Microsoft Visual Studio.*

Microsoft Visual Studio включает в себя один или несколько компонентов: Visual Basic .Net; Visual C++; Visual C#; Visual F# и др. Это позволяет разработчику выбрать наиболее удобную систему программирования для реализации определённого этапа работы.

Трансля́ция — преобразование программы, представленной на одном из языков программирования, в программу на другом языке. Язык, на котором представлена входная программа, называется исходным языком, а сама программа — **исходным кодом.** Выходной язык называется целевым языком.

Например, *программа, написанная на языке высокого уровня (исходный код), для того, чтобы ее понимал компьютер, проходит этап трансляции – преобразования в машинный код.*

Трансля́тор — программа или техническое средство, выполняющее трансляцию программы. Транслятор обычно выполняет также диагностику ошибок, формирует словари идентификаторов, выдаёт для печати текст программы и т. д.

В общем случае понятие трансляции относится не только к языкам программирования, но и к другим языкам — как формальным компьютерным (вроде языков разметки типа HTML), так обычным разговорным (русский, английский и т.п.)

Цель трансляции — преобразование текста с одного языка на язык, понятный адресату. При трансляции компьютерной программы адресатом может быть:

- устройство — процессор (трансляция называется компиляцией);
- программа — интерпретатор (трансляция называется интерпретацией).

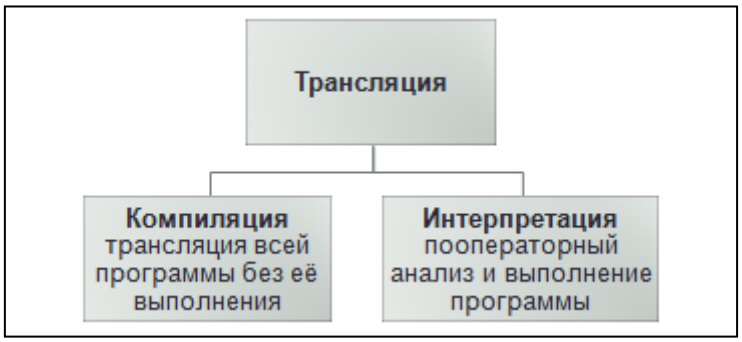
Виды трансляции:

1) компиляция:

- в исполняемый код (машинный или байт-код);
- транспиляция;

2) интерпретация;

3) динамическая компиляция.



Компилятор — программа, которая преобразует язык программы, написанной пользователем, в определенную форму, пригодную для выполнения на вычислительной машине. Основная задача здесь — перевод понятий, близких к предметной области программиста, в понятия, которыми может манипулировать персональный компьютер.

Для любого языка программирования необходим свой компилятор. При появлении нового языка появляется необходимость в переводе написанного на нем кода в вид, который сможет понять компьютер. В противном случае, код не будет выполнен. Всегда имеется семантический зазор между понятиями человека и персонального компьютера. Компиляторы языка программирования предназначены как раз для его преодоления.

Язык процессора (устройства, машины) называется машинным языком, машинным кодом. Код на машинном языке исполняется процессором. Как правило, машинный язык — язык низкого уровня. **Компилятор** — это вид транслятора, преобразующий исходный код с какого-либо языка программирования на машинный язык.

Процесс компиляции, как правило, состоит из нескольких этапов:

- лексический анализ;
- синтаксический анализ;
- семантический анализ;
- создание на основе результатов анализов промежуточного кода;
- оптимизация промежуточного кода;
- создание объектного кода, в данном случае машинного.

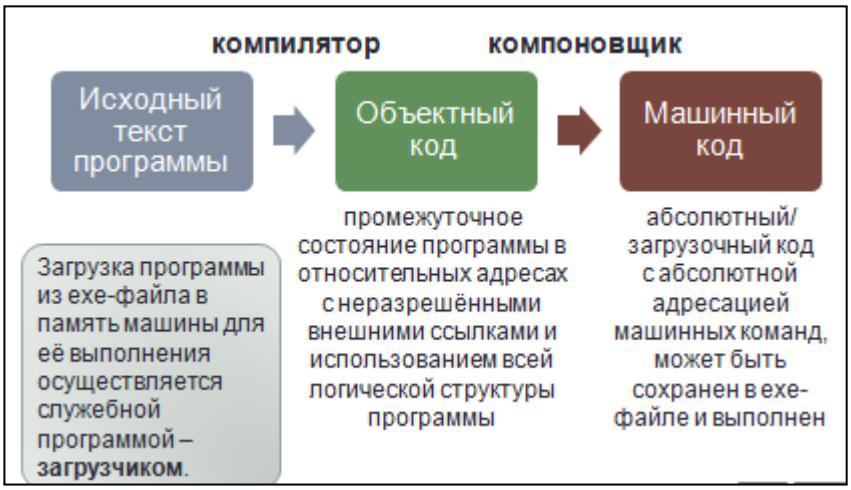
Программа может использовать сервисы, предоставляемые операционной системой, и сторонние библиотеки (например, библиотеки для работы с файлами и библиотеки для создания графического интерфейса). Для добавления в объектный файл машинного кода из других объектных файлов (кода статических библиотек) и информации о динамических библиотеках выполняется связывание (англ. *link*) или **компоновка.**

Компоновщик (редактор связей) может быть отдельной программой или частью компилятора, он создаёт исполняемый файл. Исполняемый файл (программа) запускается следующим образом:

- по запросу пользователя в ядре операционной системы создаётся объект «процесс»;
- загрузчик программ операционной системы выполняет следующие действия:

- читает исполняемый файл;
- загружает его в память;
- загружает в память динамические библиотеки;
- выполняет связывание машинного кода программы с динамическими библиотеками (динамическое связывание);
- передаёт управление программе.

Компиляция выполняется один раз и не требует наличие компилятора на устройстве, однако это медленный процесс, и при внесении изменений в исходный код требуется повторная компиляция.



Интерпретатор сразу производит анализ, перевод в машинный код и выполнение программы строка за строкой. Поэтому интерпретатор должен находиться в оперативной памяти в течение всего времени выполнения программы пользователя.

Интерпретатор может работать двумя способами:

1. читать код и исполнять его сразу без использования трансляции **(чистая интерпретация);**

2. читать код, транслировать в памяти промежуточное представление кода (байт-код или p-код), выполнять промежуточное представление кода **(смешанная реализация).**

Чистая интерпретация применяется, обычно, для языков с простой структурой, например, языков сценариев, языков APL и Лисп.

Примеры интерпретаторов, создающих байт-код: Perl, PHP, Python, Erlang.

Этапы работы интерпретатора:

- лексический анализ;
- синтаксический анализ;
- семантический анализ;
- создание промежуточного представления кода (при чистой интерпретации не выполняется);
- исполнение.

Интерпретатор моделирует машину (виртуальную машину) и реализует цикл выборки-исполнения команд машины. Команды машины записываются не на машинном языке, а на языке высокого уровня. Интерпретатор можно назвать исполнителем языка виртуальной машины.

Интерпретаторы работают в интерактивном режиме и не требуют перекомпиляции исходного кода после внесения изменений или при переносе кода на другую платформу.

Недостатки интерпретаторов по сравнению с компиляторами:

При интерпретации скорость выполнения программы существенно снижается, однако весь процесс прохождения программы на ЭВМ упрощается и имеется возможность организации диалогового (интерактивного) режима отладки и выполнения программы.

Динамическая компиляция (*JIT-компиляция*) — это трансляция, при которой исходный или промежуточный код преобразуется (компилируется) в машинный код непосредственно во время исполнения, «на лету» (англ. *just in time, JIT*). Компиляция каждого участка кода выполняется только один раз; скомпилированный код сохраняется в кэше и при необходимости используется повторно.

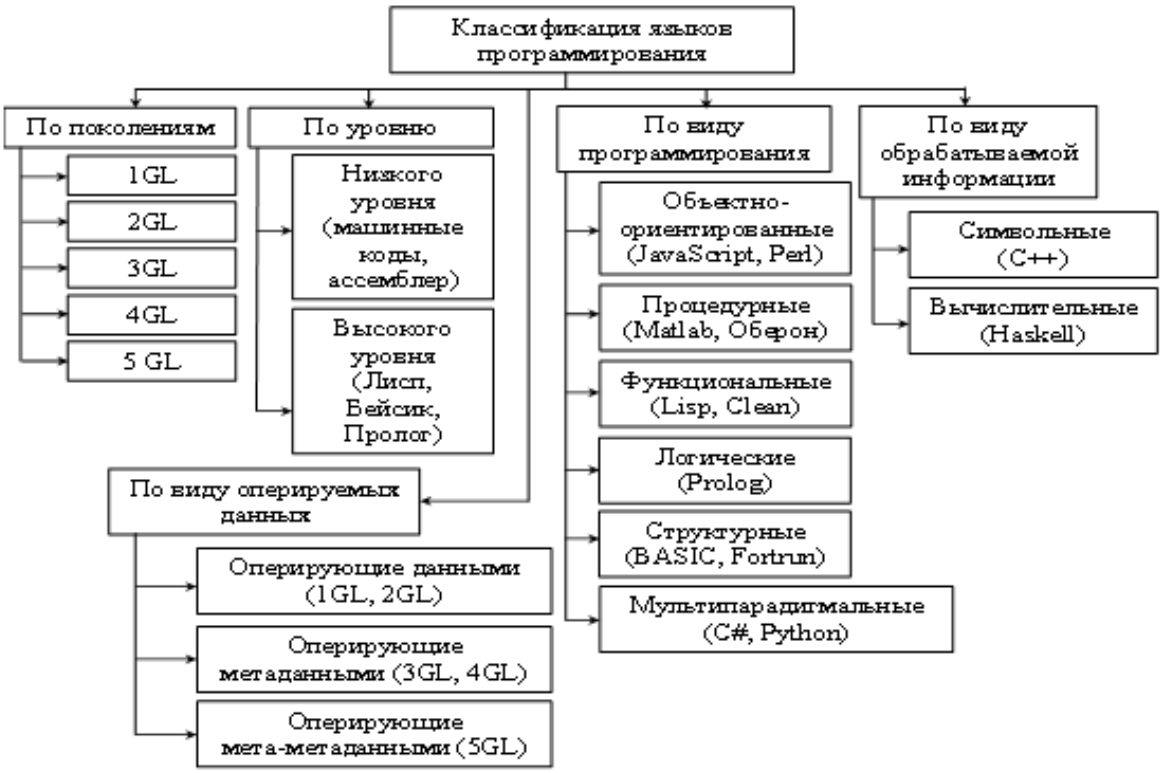
Скорость работы таких программ близка к скорости работы компилируемых, однако такие программы требуют больших ресурсов и сложны в реализации.

Динамическая компиляция появилась и поддерживается в той или иной мере в реализациях Java, .NET Framework, Perl, Python.

2. Классификация языков программирования

Язык программирования – это формальный язык, предназначенный для записи компьютерных программ.

Язык программирования определяет набор лексических, синтаксических и семантических правил, определяющих внешний вид программы и действия, которые выполнит исполнитель (обычно — ЭВМ) под её управлением.



Выбор языка/системы программирования определяется следующими факторами:

- типом решаемой задачи;
- располагаемыми вычислительными средствами;
- вкусами и знаниями заказчика и разработчика.

Различия языков по уровню:

- низкого уровня (учитывает архитектуру процессора, например, Ассемблер);
- высокого уровня **(**не учитывает архитектуру процессора, например: Паскаль, Бейсик, Си).

3. Поколения языков программирования

Языки программирования 1 поколения (1GL):

1) Машинный язык (платформенно-ориентированный код) — система команд (набор кодов операций) конкретной вычислительной машины, которая интерпретируется непосредственно процессором или микропрограммами машины. Представляет собой двоичный код, состоящий только из нулей и единиц. Иногда для упрощения программу записывают в шестнадцатеричном коде, который переводится в двоичный непосредственно микропроцессором. Машинные языки хороши для детального понимания функционирования конкретной машины, но сложны для изучения и решения прикладных задач.

Пример машинного кода:

1001 0001	91
1000 1000	88
1000 1100	8C
1000 0010	82
1000 1110	8E
1000 1011	8B

Данной программой закодировано слово «символ»

Чтобы работать с компьютерами первого поколения, программисты писали свои программы в машинных кодах, для этого им приходилось даже думать в той манере, как работают компьютеры. Программирование с помощью двоичного кода было очень медленным и трудоемким процессом, так как все операции (сложение, вычитание и т.д.) и сами данные, переводились в последовательность 0 и 1.

С развитием аппаратного обеспечения компьютеров, увеличивалась скорость обработки и емкость памяти. Это привело к изменениям в языках программирования – они стали проще и понятнее для людей. Языки программирования в своем развитии прошли практически те же стадии, что и сами компьютеры. Основная тенденция здесь – увеличение простоты взаимодействия пользователя с аппаратным и программным обеспечением компьютеров.

2) Позднее стал применяться **язык ассемблера** – язык, в котором двоичные и шестнадцатеричные коды стали заменяться буквенными обозначениями, которые называются **мнемоники**. Программа из языка ассемблера переводилась в машинный код при

помощи программы-транслятора, которая называется ассемблер (данная программа дала название языку).

Пример кода:

```
SECTION.text
org 0x100
mov ah, 0x9
mov dx, hello
int 0x21
mov ax, 0x4c00
int 0x21
SECTION.data
hello: db "Hello, world!", 0xD, 0xA.
```

Многие программисты продолжают использовать Ассемблер, так как этот язык программирования дает им полный контроль над аппаратным обеспечением компьютера и генерирует очень эффективный исполняемый код. Как и машинные коды, Ассемблер разрабатывается под определенные типы компьютеров и микропроцессоров. Недостатки этого языка в том, что программирование требует больших затрат времени, он труден для изучения и понимания; а программы, написанные на Ассемблере трудно отлаживать.

Ассемблер сегодня используется в основном для написания системного ПО.

Языки программирования 2 поколения (2GL) появились в 1950-е годы для перехода в выражениях языка от низкоуровневых машинных понятий ближе к тому, как обычно мыслит программист.

Основные отличия от языков 1 поколения:

- команды пишутся словами;
- для перевода программы в машинный код применяется программа – компилятор.

Вместе с появлением компьютеров третьего поколения, развитие языков программирования также вступило в новую фазу. Период с середины 50-х до 70-х годов отмечен появлением первых языков программирования высокого уровня (high-level languages). Для этого понадобились более быстрые, высокоэффективные компиляторы, поскольку при преобразовании исходного кода, выходные программы получались большими.

К языкам поколения 2 GL относятся: *Fortrun, Cobol и т.д.*

Пример программы на языке «Fortrun»:

```
PROGRAM PR_1
INTEGER:: J=2
REAL :: A=3.4, F=5.25, B=9.7
A=F ! значение переменной F присваивается переменной A
J=B ! значение переменной B присваивается переменной J
PRINT*, “A=”,A,” F=”,F,” J=”,J ! вывод результатов на экран
END
```

В результате выполнения программы выводится: A=5.25 F=5.25 J=9

FORTRAN – FORmula TRANslator был спроектирован в 1956 году в основном для инженеров, математиков и ученых, которые имеют в основном дело с формулами и проблемами, ориентированными на вычисления. На FORTRANe можно довольно просто

описывать сложные вычисления, манипулировать массивами и распечатывать выходные множества чисел. Хотя на этом языке было написано немало бизнес-приложений, он не очень подходит для частых операций ввода-вывода и работы со списками. FORTRAN сравнительно легко осваивается, но его синтаксис очень требователен к точности ввода операторов, что вызывает частые ошибки и делает сложной отладку программ.

Этот язык был первым, который стал широко преподаваться для обучения студентов разработке простых программ. Даже сегодня в научном и инженерном сообществе язык FORTRAN еще достаточно распространен и превалирует над многими языками.

COBOL – Common Business-Oriented Language – общий язык для приложений, ориентированных на бизнес и коммерцию. COBOL был разработан в начале 60-х годов, для того чтобы было легче писать программы для бизнеса, которые могли использоваться в таких видах деятельности как обработка заказов, ведение бухгалтерии, планирование производства и т.д. Данные, которыми оперирует COBOL – это, прежде всего, записи, файлы, таблицы и списки. Программы, написанные на языке COBOL, относительно понятны даже неспециалистам.

COBOL и FORTRAN были первыми языками, которые действительно сделали программирование доступным для обычных людей, а также позволили ученым (прежде всего, математикам) работать с компьютерами. Машинные коды и ассемблер были слишком тяжелыми для изучения и трудными в использовании.

Языки программирования 3 поколения (3GL) унаследовали все достоинства языков 2 поколения и дополнили их своими:

1) Простота и понятность использования.

2) Независимость от конкретного компьютера – это достигалось тем что теперь между пользователем и архитектурой ЭВМ была ОС.

3) Возможность использования специальных синтаксических приемов – программы стали более сложными в них появились блоки команд объединённые в процедуры и функции.

4)Модульность программ – написание отдельных процедур и функций для решения отдельных маленьких задач позволило их повторно использовать в других проектах.

Пример программы на языке Pascal:

```
Данная программа вводит с клавиатуры n чисел и определяет четные.
-----------------------------------------------
var n, i, k, a: integer;
begin
writeln ('введите количество чисел');
Readln (n);
a:=0;
for i:=1 to n do begin
      writeln ('введите ',i:1,'-е число');
      readln (a);
      if a mod 2=0 then k:=k+1;
      end;
writeln ('кол-во четных чисел ',k);
readln;
end.
```

Примеры языков программирования 3GL: *Lisp, BASIC, Pascal, C и т.д.*

BASIC (Beginners All-purpose Symbolic Instruction Code) был создан в 1964 году для обучения студентов колледжей использованию компьютеров. Сегодня это один из самых используемых языков программирования. Это простой язык, который недавно подошел к такому уровню развития, при котором его можно использовать даже для построения больших систем высокой производительности. Слабая сторона BASIC – то что он выполняет все задачи одинаково, без оптимизации кода. А ещё ему очень не хватает формальной структуры, которая делает язык, подобный C и C++, более подходящим для больших проектов.

Названный в честь Блеза Паскаля, французского математика и философа, язык программирования **Pascal** был разработан швейцарским профессором компьютерных наук Никлаусом Виртом в конце 60-х. Программирование на Pascal стало весьма популярным на платформе микрокомпьютеров, так как программы, написанные на нем занимают немного места в памяти. Pascal сочетает в себе возможности обработки сложных массивов данных, набор простых, но мощных команд и высокоэффективный выходной код. Поэтому его часто применяют для обучения будущих профессионалов навыкам программирования. Для начинающих Pascal слишком сложен, а профессионалы для разработки сложных приложений предпочитают C. Pascal

Языки программирования 4 поколения (4GL) относятся к временному периоду с 1980-х настоящее время и предназначены для реализации крупных проектов, повышают их надежность и скорость создания, ориентированы на специализированные области применения. В них встроены операторы, позволяющие одной строкой описать такую функциональность, для реализации которой на языках младших поколений потребовались бы тысячи строк исходного кода.

Данные языки наряду с языками 3GL оперируют **метаданными** (данные о данных, раскрывающие сведения о признаках и свойствах, характеризующих какие-либо сущности, позволяющие автоматически искать и управлять ими в больших информационных потоках).

Примеры языков программирования 4GL: C#, 1C, JavaScript, SQL, Prolog, Python.

Пример программы нахождения 10 наиболее частых слов на web-странице (язык «Python»):

```
from urllib2 import urlopen
u = urlopen("http://python.org")
words = {}
for line in u:
   line = line.strip(" \n")
   for word in line.split(" "):
      try:
         words[word] += 1
      except KeyError:
         words[word] = 1
pairs = words.items()
pairs.sort(key=lambda x: x[1],
reverse=True
for p in pairs[:10]:
   print(p[0], p[1])
```

Четвертое поколение языков программирования зародилось в конце 70-х, а развитие их продолжается по сей день. Эти языки существенно уменьшили время разработки ПО и

позволили выполнять эту работу даже людям без технического образования. Сегодня для выполнения многих задач программирование как таковое вообще не требуется. Например, появление приложений электронных таблиц (spreadsheets), таких как Microsoft Excel, позволяет обычным пользователям обрабатывать финансовую информацию и управлять большими массивами данных. До 70-х годов без применения языков программирования использовать возможности компьютеров было невозможно.

Язык С (читается "си") был разработан в начале 70-х в AT&T Bell Labs. На сегодня эти языки являются фактически единственным выбором для построения операционных систем и сложных приложений, таких как электронные таблицы, компиляторы, сетевые утилиты, коммерческие приложения и проч. Операционные системы UNIX, Linux и Windows большей частью написаны на С, даже компилятор С написан на С. Эти мощные и сложные языки генерируют быстрый и эффективный код. Работать на С можно на любых компьютерах – от микро ЭВМ до мэйнфреймов. Программист, владеющий С, имеет полный контроль над средой разработки и может заставить компьютер делать практически все, что ему нужно. С сильно потеснил COBOL в приложениях для бизнеса, но основная область его применения – коммерческие пакеты прикладных программ для микрокомпьютеров – серверов, рабочих станций и ПК. Как и ожидалось, эти языки очень сложны, и для того, чтобы овладеть навыками программирования на С, одного желания мало – нужен талант, как и в случае работы с Ассемблером и машинными кодами.

Языки программирования 5 поколения (5GL)

Планируются в будущем. В настоящее время данных языков программирования не существует. Предполагается, что они:

1. Будут оперировать **мета-мета-данными** (сведения о сведениях о данных, т.е. содержат конкретные количественные и качественные показатели признаков и свойств о данных);
2. Будут иметь производительность в 10-10000 раз более высокую по сравнению с языками программирования 3GL и 4GL.
3. Возможность автоматического формирования результирующего текста на универсальных языках программирования, ввод инструкций в максимально наглядном виде с помощью методов, наиболее удобных для человека, не знакомого с программированием.

Сейчас существует единственный язык, который работает с мета-мета-данными, - это язык команд менеджеров пакетов или менеджеров зависимостей, таких как apt, yum, smart, maven, cpan и другие. Однако данный язык является языком командной строки, а не языком программирования).

В компьютерном мире было, по крайней мере, две неудавшиеся попытки разработать и внедрить некий универсальный язык. Первая, в середине 60-х, ознаменовалась изобретением IBM языка, названного PL/1. Не так давно Министерство обороны США постановило разработать язык, который должен был затем получить статус стандарта. Это усилие было не более успешным, чем попытка IBM продвинуть PL/1, заменяющий сотни языков программирования, широко используемых во всём мире.

4. Языки определения и манипулирования данными

На современном этапе развития информационных технологий достигнуты большие успехи в получении, обработке, хранении, управлении, анализе и визуализации данных. Совместно эти задачи называются **управлением данными.**

Для хранения и обработки данных можно использовать либо электронные таблицы, либо системы управления базами данных. Использование систем управления базами данных требует более высокого уровня подготовки для создания таблицы и описания обработки ее данных.

В истории вычислительной техники можно проследить развитие двух основных областей ее использования:

1) применение вычислительной техники для выполнения численных расчетов;
2) использование средств вычислительной техники в автоматических или автоматизированных информационных системах.

Информационные системы имеют дело с большими объемами информации, имеющей достаточно сложную структуру (банковские системы, автоматизированные системы управления предприятиями, системы резервирования авиационных и железнодорожных билетов, мест в гостиницах и т.д.). Современные информационные системы характеризуются огромными объемами хранимых данных, сложной организацией, необходимостью удовлетворять разнообразные требования многочисленных пользователей.

Разработчики информационных систем предложили новый подход к управлению информацией. Этот подход был реализован в рамках новых программных систем, названных впоследствии **Системами Управления Базами Данных (СУБД)**, а сами хранилища информации, которые разработаны под управлением данных систем, назывались базами или банками данных (БД и БнД).

Основные требования, предъявляемые к банкам данных:

- многократное использование данных,
- простота,
- легкость использования,
- гибкость использования,
- быстрая обработка запросов на данные,
- язык взаимодействия.

Основой информационной системы является база данных.

Теория баз данных — сравнительно молодая область знаний. Возраст ее составляет немногим более 45 лет.

База данных (БД) - это именованная совокупность структурированных данных, отображающая состояние объектов и их отношения в рассматриваемой предметной области. База данных лежит в основе каждой информационной системы. А целью любой информационной системы является обработка данных об объектах реального мира.

Предметная область БД — часть реального мира, подлежащего изучению. Структурирование БД — введение соглашений о способах представления данных. Пользователи БД: различные прикладные программы, специалисты предметной области, выступающие в роли потребителей или источников данных (конечные пользователи), администраторы баз данных.

Система управления базами данных (СУБД) — это комплекс программных и языковых средств, необходимых для создания для создания, ведения и совместного использования БД многими пользователями, поддержания БД в актуальном состоянии.

Это программная система, предназначенная для создания на ЭВМ общей БД, используемой для решения множества задач. СУБД представляет собой оболочку, с помощью которой при организации структуры таблиц и заполнения их данными получается та или иная база данных. В связи с этим в составе СУБД различают систему программно-технических, организационных и "человеческих" составляющих. Программные средства

включают систему управления, обеспечивающую ввод-вывод, обработку и хранение информации, создание, модификацию и тестирование БД, трансляторы.

Исторически для системы управления базой данных сложились три языка:
1) Язык описания данных, называемый также языком описания схем, - для построения структуры ("шапки") таблиц БД.
2) **Язык манипулирования данными** - для заполнения БД данными и операций обновления (запись, удаление, модификация);
3) **Язык запросов** - язык поиска наборов величин в файле в соответствии с заданной совокупностью критериев поиска и выдачи затребованных данных без изменения содержимого файлов и БД (язык преобразования критериев в систему команд).

В настоящее время функции всех трех языков выполняет язык SQL, относящийся к классу языков, базирующихся на исчислении кортежей (кортеж - единица информации), языки СУБД FoxPro, Visual Basic for Application (СУБД Access) и т.д.

Базовыми внутренними языками программирования являются языки четвертого поколения. В качестве базовых языков могут использоваться C, C++, Pascal, Object Pascal.

SQL (Structured Query Language - Структурированный язык запросов) — язык управления базами данных для реляционных баз данных. Сам по себе SQL не является полным языком программирования, но его стандарт позволяет создавать для него процедурные расширения, которые расширяют его функциональность до полноценного языка программирования.

Первый официальный стандарт языка был принят ANSI в 1986 году и ISO в 1987. С тех пор были созданы еще несколько версий стандарта, некоторые из них повторяли предыдущие с незначительными вариациями, другие принимали новые существенные черты.

Несмотря на существование стандартов, большинство распространенных реализаций SQL отличаются так сильно, что код редко может быть перенесен из одной системы управления базами данных в другую без внесения существенных изменений. Это объясняется большим объемом и сложностью стандарта, а также нехваткой в нем спецификаций в некоторых важных областях реализации.

SQL создавался как простой стандартизированный способ извлечения и управления данными, содержащимися в реляционной базе данных. Позднее он стал сложнее, чем задумывался, и превратился в инструмент разработчика, а не конечного пользователя. В настоящее время SQL (по большей части в реализации Oracle) остается самым популярным из языков управления базами данных, хотя и существует ряд альтернатив.

SQL состоит из четырех отдельных частей:

1**. Язык определения данных (DDL)** используется для определения структур данных, хранящихся в базе данных. Операторы DDL позволяют создавать, изменять и удалять отдельные объекты в базе данных. Допустимые типы объектов зависят от используемой системы управления базами данных и обычно включают базы данных, пользователей, таблицы и ряд более мелких вспомогательных объектов, например, роли и индексы.

2. **Язык манипуляции данными (DML)** используется для извлечения и изменения данных в базе данных. Операторы DML позволяют извлекать, вставлять, изменять и удалять данные в таблицах. Иногда операторы select извлечения данных не рассматриваются как часть DML, поскольку они не изменяют состояние данных. Все операторы DML носят декларативный характер

3. **Язык определения доступа к данным (DCL)** используется для контроля доступа к данным в базе данных. Операторы DCL применяются к привилегиям и позволяют выдавать и отбирать права на применение определенных операторов DDL и DML к определенным объектам базы данных.

4. **Язык управления транзакциями (TCL)** используется для контроля обработки транзакций в базе данных. Обычно операторы TCL включают commit для подтверждения изменений, сделанных в ходе транзакции, rollback для их отмены и savepoint для разбиения транзакции на несколько меньших частей.

SQL реализует декларативную парадигму программирования: каждый оператор описывает только необходимое действие, а система управления базами данных принимает решение о том, как его выполнить, т.е. планирует элементарные операции, необходимые для выполнения действия и выполняет их..

Литература:

1. Лягинова О.Ю., Можаева М.Г. Этапы решения задач на ЭВМ: онлайн-презентация / Кафедра математики и информатики ЧГУ / https://en.ppt-online.org/593552 (стр. 16-23)
2. Онлайн-презентация / https://en.ppt-online.org/550729
3. https://thepresentation.ru/uncategorized/bazy-dannyh-sistemy-upravleniya-bazami-dannyh
4. https://studopedia.org/14-75717.html
5. https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%BB%D1%8F%D1%82%D0%BE%D1%80
6. Участие в интеграции программных модулей: отчет по производственной практике / https://knowledge.allbest.ru/programming/2c0b65625a2bc79a4c53b89521306d36_0.html#text