Билет 1

1) **Простейшая вариационная задача. Необходимое условие экстремума функционала. Уравнение Эйлера. Методы решения.**

Задачи, в которых необходимо найти оптимальную функцию, называются вариационными.

Оператор, ставящий в соответствие некоторой функции число, называется функционалом:

$$I = I[x(t)]$$

Простейшая вариационная задача:

Пусть дан функционал вида $I = \int_{t_2}^{t_1} f\big(t, x(t), x'(t)\big)dt$ и заданы краевые условия:

$x(t_0) = x_0$

$x(t_1) = x_1$

Необходимо найти функцию $x^*(t)$, удовлетворяющую краевым условиям, при которой функционал достигает минимума или максимума.

Необходимое условие экстремума функционала:

Если $x^*(t)$ доставляет экстремум функционалу, то $\partial I[x^*(t)] = 0$

Используя необходимое условие экстремума функционала легко показать, что $x^*(t)$ является решением уравнения Эйлера следующего вида:

$$\frac{\partial f}{\partial x} = \frac{d}{dt}\left(\frac{\partial f}{\partial x'}\right) = 0$$

Таким образом последовательность решения простейшей вариационной задачи заключается в получении и решении уравнения Эйлера. Найденная функция называется экстремалью. Найденная экстремаль является лишь претендентом на решение исходной задачи, необходимо проверить достаточное условие.

Методы решения:

1) Численные методы решения уравнения Эйлера (метод пристрелки, метод пристрелки для уравнения Эйлера-Пуассона, метод прогонки)
2) Прямые методы решения вариационных задач (метод Ритца, метод Конторовича, конечно-разностный метод Эйлера)

2) **Геометрические преобразования в трехмерной графике. Матрицы преобразования.**

При построении изображений часто приходится иметь дело с ситуациями, когда общее изображение (рисунок) включает в себя целый ряд компонент (подрисунков), отличающихся друг от друга только местоположением, ориентацией, масштабом, т.е. отдельные подрисунки обладают значительным геометрическим сходством.

В этом случае целесообразно описать один подрисунок в качестве базового, а затем получать остальные требуемые подрисунки путем использования операций преобразования.

Матрицы преобразования:

1) Перемещение

$$T(D_x, D_y, D_z) = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ D_x & D_y & D_z & 1 \end{vmatrix}$$

2) Масштабирование

$$S(S_x, S_y, S_z) = \begin{vmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

если Sx = Sy = Sz, то это однородное масштабирование.

3) Поворот

Относительно оси Z:

$$R_z(\alpha) = \begin{vmatrix} cos\alpha & sin\alpha & 0 & 0 \\ -sin\alpha & cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

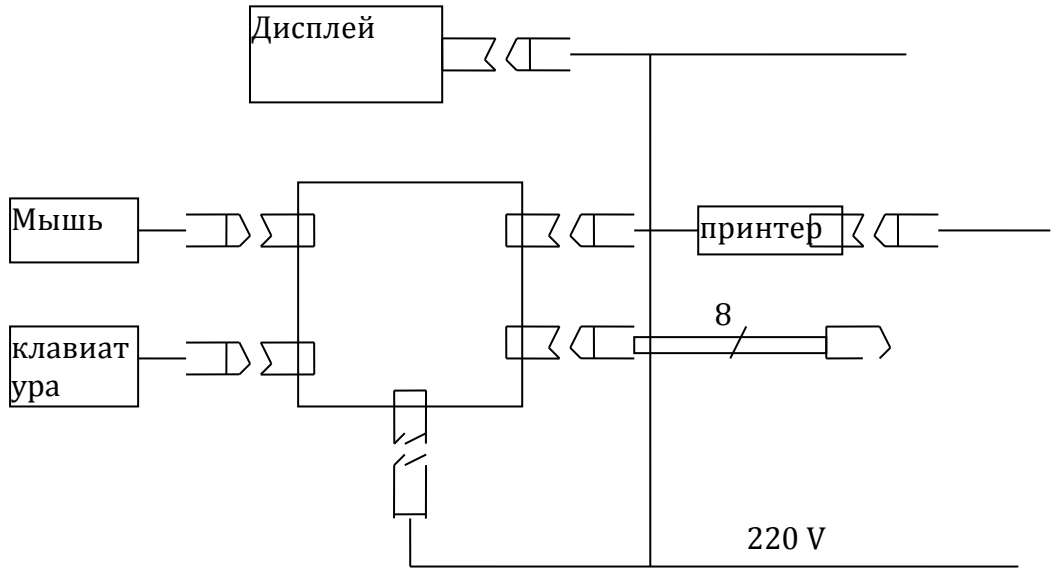
Относительно оси X:

$$R_x(\alpha) = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\alpha & sin\alpha & 0 \\ 0 & -sin\alpha & cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

Относительно оси Y:

$$R_y(\alpha) = \begin{vmatrix} cos\alpha & 0 & sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ sin\alpha & 0 & cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

3) Составить электрическую схему автоматизированного рабочего места инженера на базе ПЭВМ.



Билет 2

1) **Трехмерная графика. Методы удаления скрытых поверхностей, использующие Z-буфер.**

- Алгоритм удаления невидимых граней, использующий Z-буфер

Для реализации этого алгоритма требуется два буфера:

1) Буфер глубины (*Z* - буфер).
2) Буфер регенерации.

Из трехмерной сцены выбираем последовательно грани и развертываем в растр. Но предварительно буфер регенерации заполняем фоновым цветом, а в *Z* - буфер помещаем значения максимально большие для этой сцены. В *Z* - буфере получаются значения значительно большие чем глубина сцены. Для каждого многоугольника во время растровой развертки выполняем следующие алгоритмические шаги:

1) Если глубина многоугольника Z(x, y) в текущей точке растровой развертки меньше чем соответствующая точка в *Z* - буфере, то точка находится ближе к наблюдателю и в буфер регенерации в точку (x, y) записываем атрибут многоугольника, $Z_{БУФ}(x, y) <- Z(x, y)$.
2) Иначе переход к следующей точке растровой развертки многоугольника.

Главным недостатком алгоритма является большой размер *Z* - буфера. Сцена будет появляться в той последовательности в какой мы анализируем грани.
Достоинства: обрабатываются сцены любой сложности, прост в реализации.

- Алгоритм удаления невидимых граней, использующий Z-строку

Работает в рамках одной сканирующей строки. Количество элементов в Z - строке соответствует разрешающей способности по горизонтали. Глубина Z - строки определяет величину значения Z (см. Алгоритм использующий Z - буфер). Для повышения эффективности работы алгоритма за каждым многоугольником закрепляют верхнюю и нижнюю сканирующие строки.

2) **Использование булевой алгебры для анализа и синтеза логических электронных схем.**

Функциональную схему логического устройства получают в результате абстрактного синтеза, который состоит из следующих этапов:

1. словесная формулировка функций логического устройства
2. составление таблицы истинности по словесной формулировке
3. запись логического уравнения устройства в виде СДНФ или СКНФ
4. минимизация логического уравнения
5. выбор одного из логических базисов
6. преобразование логического уравнения с использованием правил де Моргана
7. построение функциональной схемы логического устройства

Пример:

1. синтезировать логическое устройство на три входные переменные генерирующее сигнал 1 на выходе, если две рядом стоящие переменные из трех принимают значение 1
2. таблица истинности

A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0

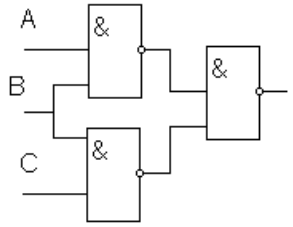
1	1	0	1
1	1	1	1

3. $Y = \overline{A} \cap B \cap C \cup A \cap B \cap \overline{C} \cup A \cap B \cap C$

4. $Y = \overline{A} \cap B \cap C \cup A \cap B \cap (C \cup \overline{C}) = \overline{A} \cap B \cap C \cup A \cap B = B \cap (\overline{A} \cap C \cup A) = B \cap (A \cup C) =$
$= B \cap A \cup B \cap C$

5. принять для реализации схемы логического устройства базис и-не

6. $Y = \overline{\overline{Y}} = \overline{\overline{\overline{(B \cap A \cup B \cap C)}}} = \overline{(\overline{(B \cap A)} \cap \overline{(B \cap C)})}$



3) Найти кратчайшее расстояние от точки A(0;1) до прямой y=2x+3, используя методы вариационного исчисления.

1) Аналитический

y=2x+3 -> 2x-y+3=0, тогда нормальный вектор к данной прямой n={2: -1}
пусть P – основание перпендикуляра , тогда уравнение прямой PA, перпендикулярной исходной прямой будет иметь вид

$$\frac{x-0}{2} = \frac{y-1}{-1}$$

Определим координаты точки P
т.к. P точка пересечения двух прямых решив систему, найдем ее координаты

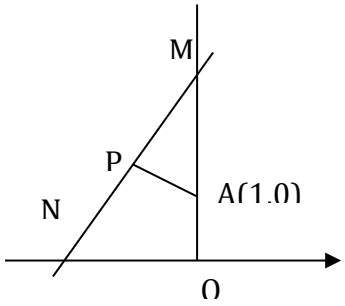
{ x/2+y=1
{ y=2x+3

{ x=-4/5
{ y=7/5

теперь найдем искомое расстояние AP

AP= $\sqrt{(0+4/5)^2 + (1-7/5)^2} = 0{,}4\sqrt{5}$

2) Геометрический



AP(искомое расстояние) перпендикулярно MN
тр-ик MNO подобен тр-ику MPA -> MN/MA = NO/AP -> AP=NO*MA/MN

MA=2 NO=1.5 MN= $\sqrt{MO^2 + NO^2} = \sqrt{3^2 + 1.5^2} = 1.5\sqrt{5}$

AP=0.4 $\sqrt{5}$

3) Оптимизационный

Запишем функцию расстояния от точки до прямой и любым методом оптимизации (например, сканирование, метод золотого сечения)
y=2*x+3
s=sqrt(x^2+(2x+3-1)^2)=sqrt(x^2+4*(x+1)^2)=sqrt(5*x^2+8*x+4);
s'=(10*x+8)*0.5/ sqrt(5*x^2+8*x+4)= (5*x+4)/sqrt(5*x^2+8*x+4)=0; следовательно x=-0.8;
s=sqrt(0.8)= 0.4 $\sqrt{5}$

Билет 3

1) Виды рисков сбоя в асинхронных комбинационных схемах.

Риском сбоя называется возможность появления на выходе цифрового устройства сигнала, не предусмотренного алгоритмом его работы и могущего привести к ложному срабатыванию.

Состязаниями (гонками) сигналов называется процесс их распространения в различных цепях цифрового устройства при существовании разбросов временных задержек этих цепей.

Виды рисков сбоя:

1) Динамический

Риск сбоя называется динамическим, если состояние выхода **y** от комбинации входных сигналов X_1, в первый момент, отличается от состояния выхода **y** при изменении комбинации входных сигналов X_2, в следующий момент времени, где **y** - булева функция.
Риск сбоя называется динамическим D_+ при изменении выходного значения 0→1, если $y(X_1) = 0$, а $y(X_2) = 1$.
Риск сбоя называется динамическим D_-, если $y(X_1) = 1$, а $y(X_2) = 0$.
Динамический риск сбоя является следствием статического риска сбоя. Динамические риски сбоя в цифровой схеме могут привести к нарушению закона ее функционирования.

2) Статический

Риск сбоя называется статическим, если $y(X_1) = y(X_2)$, где y - булева функция. Риск сбоя называется статическим в нуле S_0, если $y(X_1) = y(X_2) = 0$. Риск сбоя называется статическим в единице S_1, если $y(X_1) = y(X_2) = 1$.

3) Функциональный

Риски сбоя, проявляющиеся при многоместной смене наборов и определяемые характером самой функции, называются функциональными. Такие риски сбоя не могут быть устранены изменением логической структуры, реализующей булеву функцию.

4) Логический

Статический риск сбоя в комбинационной схеме бывает логическим. Статический риск сбоя называют логическим, если он обусловлен выбранной схемной реализацией функции. Логический риск сбоя при смене входных воздействий может быть исключен модификацией схемы. Формальное основание для подобных действий дает следующее утверждение: Необходимым и

достаточным условием исключения логического риска сбоя является реализация всех простых импликант функции в схеме.

2) Понятие телеобработки. Терминальная и системная телеобработка.

Телеобработка данных — это такая организация информационно-вычислительного процесса, при которой ресурсы одной или нескольких ЭВМ одновременно используются многими пользователями через различные виды средств связи.

Системы телеобработки обеспечивают организацию двух основных методов обработки данных — пакетного и диалогового. Соответственно выделяются и два основных режима взаимодействия удаленного абонента с ЭВМ — режим пакетной передачи данных и диалоговый режим взаимодействия.

Системная телеобработка данных

Системная телеобработка определяется как взаимоувязанная совокупность технических и программных средств, обеспечивающая коллективное использование ресурсов и баз данных одной ЭВМ (вычислительного комплекса) большим количеством пользователей, подключенных к ЭВМ через средства связи и передачи данных. Основными средствами системной телеобработки данных являются телекоммуникационные методы доступа, мультиплексоры передачи данных, абонентские пункты, аппаратура передачи данных.

Сетевая телеобработка данных

Сетевая телеобработка определяется как взаимоувязанная совокупность унифицированных логических и физических средств, протоколов, интерфейсов, обеспечивающая возможность распределения управляющих и обрабатывающих мощностей, а также баз данных по сети. Такая телеобработка обеспечивает коллективное использование ресурсов и баз данных одной или нескольких территориально рассредоточенных ЭВМ большим количеством пользователей, подключенных к ЭВМ через средства связи и передачи данных.

Для описания взаимодействия компонентов в сети используются *протоколы и интерфейсы*.

Наиболее важными функциями протоколов на всех уровнях сетевой телеобработки являются защита от ошибок, управление потоками данных в сети, защита сети от перегрузки и выполнение операций по маршрутизации сообщений и оптимизации использования ресурсов в сети.

3) Выявить сходимость метода простых итераций для решения следующих уравнений и систем:

1) $1-x^2=0$
2) $\cos(y-1)+x = 0{,}8$
   $y - \cos(x) = 2$
3) $AX = B$, где

$$A=\begin{pmatrix} 0{,}25 & -0{,}52 & 0{,}043 & 0 \\ 0{,}14 & -0{,}16 & -0{,}316 & 0 \\ 0{,}12 & 0{,}08 & -0{,}14 & -0{,}24 \\ 0{,}12 & -0{,}35 & -0{,}18 & 0 \end{pmatrix} \quad B=\begin{pmatrix} 0{,}44 \\ 1{,}42 \\ -0{,}83 \\ -1{,}42 \end{pmatrix}$$

Решение:

1) $x = \varphi(x)$ или исходное уравнение 1 преобразуется в вид $x = 1 + x - x^2$.
$\varphi'(x) = 1 - 2 \bullet x$. Для |x|<1 $\varphi'(x) < 1$, значит метод сходится для корней в окрестности точки 0.

2)Приведем данную систему к стандартному виду $\begin{cases} x = 0.8 - \cos(y-1) \\ y = 2 + \cos(x) \end{cases}$

$\varphi'(x,y)=\begin{pmatrix}\frac{\partial\varphi_1}{\partial x} & \frac{\partial\varphi_1}{\partial y}\\ \frac{\partial\varphi_2}{\partial x} & \frac{\partial\varphi_2}{\partial y}\end{pmatrix}=\begin{pmatrix}0 & \sin(y)\\ -\sin(x) & 0\end{pmatrix}; |\varphi'(x,y)| = -\sin(x)*\sin(y) <= 1$. Следовательно метод простых итераций сходится.

3) в матричном виде общая формула имеет вид x=B*x+c. Мы имеем вид A*x=b. Преобразуем: A*x+E*x=b+x или x=(A+E)*X-b.

$$c=\begin{pmatrix}-0.44\\ -1.42\\ 0.83\\ 1.42\end{pmatrix} \quad B=\begin{pmatrix}1.2500 & -0.5200 & 0.0430 & 0\\ 0.1400 & 0.8400 & -0.3160 & 0\\ 0.1200 & -0.3500 & 0.8200 & -0.2400\\ 0.1200 & -0.3500 & -0.1800 & 1.0000\end{pmatrix}$$

$|B| = 0.7172 < 1$, значит метод сходится для данной системы линейных уравнений

Билет 4

1) **Структура и принципы работы экспертной системы.**

Экспертная система - это система ИИ, созданная для решения задач в конкретной проблемной области.

Структура и состав экспертной системы:

Типовая экспертная система включает в себя:

1. Базу знаний (БЗ).
2. Механизм логического вывода (МЛВ).
3. Модуль извлечения знаний.
4. Систему объяснений.

Структура базы знаний:

База знаний включает факты и правила, по которым в зависимости от входной информации, принимается то или иное решение. Факты представляют краткосрочную информацию. Они могут не быть в БЗ и могут изменяться. Правила - долгосрочная информация о том, как манипулировать знаниями для получения решений.

Механизм логического вывода:

МЛВ решает две задачи:

1) Дополнение и изменение БЗ на основе анализа БЗ и исходной информации.
2) Управление порядком обработки БЗ.

МЛВ функционирует циклически. В каждом цикле решаются следующие задачи:

- *Сопоставление*. Сопоставляется имеющаяся условная часть с имеющимися фактами.
- *Выбор*. В случае наличия нескольких правил с одинаковой условной частью выбирается одно правило для срабатывания действия.
- *Действие*. Выполняется действие, определённое следственной частью сработанного правила.

В настоящее время существуют две основные стратегии логического вывода:

- прямая цепочка рассуждений (основана на сопоставлении исходных данных со всеми правилами в БД и получении некоторого результата);
- обратная цепочка рассуждений (выдвигается гипотеза о предполагаемом решении задачи и путём анализа БЗ ищется подтверждение этой гипотезы путём сравнения БЗ и исходных данных; если подтверждение не найдено, то выдвигается новая гипотеза).

Модуль извлечения знаний:

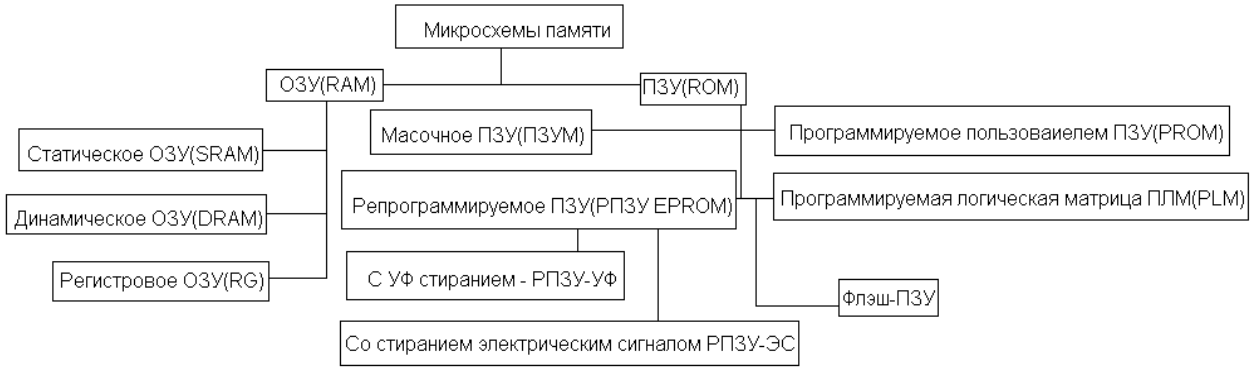
Его назначение - предоставление экспертных знаний и их структурирование в виде, пригодном для использования в компьютерпной системе.

Система объяснения:

Предназначена для показа пользователю всего процесса рассуждения, в результате которого было найдено или не найдено решение.

2) Принципы организации памяти: оперативные, постоянные запоминающие устройства. Схемотехника динамической памяти.

Современная классификация микросхем памяти:



ПЗУ и их разновидности, например программируемые логические матрицы (ПЛМ), широко используются для построения управляющих программных или микропрограммных памятей и различных логических комбинационных схем ЭВМ и систем автоматики, например преобразователей кодов, дешифраторов, генераторов последовательностей сигналов, мультиплексоров, сдвиговых и счетных регистров и т. д.

ОЗУ

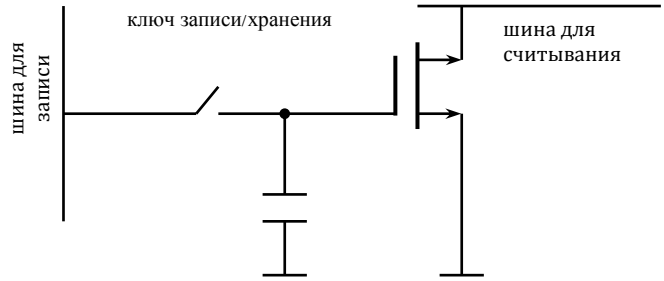
Благодаря низкой стоимости на бит и высокой плотности размещения ее элементов, динамические ОЗУ на базе БИС МОП стали доминировать в ОЗУ ЭВМ. Тем не менее существуют объективные ограничения для дальнейшего совершенствования динамических ОЗУ.

Основным ограничением динамического ОЗУ является его производительность, которая включает несколько важнейших аспектов – задержку доступа и длительность цикла доступа к строке и скорость передачи данных при доступе к столбцу.

Следующим объективным ограничением является проблема обновления памяти, что характерно только для динамических ОЗУ. Утечки заряда в элементе памяти требуют его восстановления через определенные промежутки времени (обычно от 16 до 32 мс).

Динамические ЗУ

Динамическими они называются потому, что они в принципе не способны сохранять записанную информацию длительное время и требуют периодической регенерации (обновления) хранимой информации.



При замкнутом ключе записи/хранения высокий потенциал на шине записи обеспечивает заряд емкости С (в

качестве С может использоваться и входная емкость МДП транзистора). При выключенном ключе заряд на емкости сохраняется некоторое время (единицы – сотни миллисекунд) за счет малых токов утечки через ключ записи/хранения.

МДП транзистор изолирует С от шины чтения и позволяет считать информацию без разрешения. Это важная особенность.

Наличие утечек приводит к необходимости периодической регенерации заряда на емкости С. Процедура регенерации включает последовательные чтение и запись.

3) Составить программу для вычисления скорости передачи информации между компьютерами, объединенными в локальную сеть.

```
//ПРОГРАММА СЕРВЕР
unit NetTestSrv;

interface

type
  TForm1 = class(TForm)
    Socket1: TServerSocket;
    procedure Socket1Read(Sender: TObject; Socket: TCustomWinSocket);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

type

implementation

procedure TForm1._FORM_CREATE(Sender: TObject);
begin
     Socket1.Port:=1203038;
     Socket1.Active:=True;
end;

procedure TForm1.Socket1Read(Sender: TObject; Socket: TCustomWinSocket);
begin
     Socket1.Socket.SendText(Socket.ReceiveText;);
end;

end.
//ПРОГРАММА КЛИЕНТ
unit NetTestClient;

interface
type
  TForm1 = class(TForm)
    Socket1: TClientSocket;
    procedure Socket1Read(Sender: TObject; Socket: TCustomWinSocket);
  end;

implementation

var i:integer;

procedure TForm1._FORM_CREATE(Sender: TObject);
begin
     Socket1.Address:='127.0.0.0';
     Socket1.Port:=530262;
     Socket1.Active:=True;
     i=GetTickCount();
     Socket1.Socket.SendText('TEST TEXT');
end;
```

```
end;

procedure TForm1.Socket1Read(Sender: TObject; Socket: TCustomWinSocket);
begin
    if Socket1.Socket.ReceiveText='TEST TEXT' then begin
        ShowMessage('Время передачи данных - ' + IntToStr(GetTickCount()-i) + ' мс');
    end;
end;
```

Билет 5

1) Целочисленные оптимизационные задачи и методы их решения.

Найти $x_1^*, \dots, x_n^*$, при которых $f_0(x_1, \dots, x_n) \to min$, при ограничениях:
$f_i(x_1, \dots, x_n) \geq 0, i = 1 \dots m$

Дополнительное условие:

1) $x_1, \dots, x_k$, где $k < n$ - целые (задача частично целочисленного программирования)
2) $x_i, i = 1 \dots n$ - целые и могут принимать только значения 0 и 1 (задача бивалентного программирования)

Целевая функция может быть любой.

Метод, заключающийся в решении данной задачи без учета целочисленности x_i с последующим округлением до ближайшего целого не применим.

Пример: задача о рюкзаке

Методы решения:

1) метод полного перебора
2) метод случайного поиска
3) метод ветвей и границ

2) Открытые вычислительные сетевые структуры. Эталонная модель.

С целью стандартизации различного сетевого оборудования Международной организацией по стандартизации (ISO) разработана эталонная модель взаимосвязи открытых систем (OSI), которая определяет функции и уровневую организацию набора протоколов таким образом, чтобы могло совместно работать оборудование, разработанное независимо различными фирмами, но в соответствии с одним и тем же архитектурным стандартом.

Эталонная модель OSI

Эта модель содержит в себе по сути 2 различных модели:

- горизонтальную модель на базе протоколов, обеспечивающую механизм взаимодействия программ и процессов на различных машинах
- вертикальную модель на основе услуг, обеспечиваемых соседними уровнями друг другу на одной машине

В горизонтальной модели двум программам требуется общий протокол для обмена данными. В вертикальной - соседние уровни обмениваются данными с использованием интерфейсов API.

Уровень 1, физический

Физический уровень получает пакеты данных от вышележащего канального уровня и преобразует их в оптические или электрические сигналы, соответствующие 0 и 1 бинарного потока. Эти сигналы посылаются через среду передачи на приемный узел. К числу наиболее распространенных спецификаций физического уровня относятся:

- IEEE 802.3 -- Ethernet

- IEEE 802.5 -- Token ring

Уровень 2, канальный
Канальный уровень обеспечивает создание, передачу и прием кадров данных. Этот уровень обслуживает запросы сетевого уровня и использует сервис физического уровня для приема и передачи пакетов. Спецификации IEEE 802.x делят канальный уровень на два подуровня: управление логическим каналом (LLC) и управление доступом к среде (MAC). LLC обеспечивает обслуживание сетевого уровня, а подуровень MAC регулирует доступ к разделяемой физической среде.

Уровень 3, сетевой
Сетевой уровень отвечает за деление пользователей на группы. На этом уровне происходит маршрутизация пакетов на основе преобразования MAC-адресов в сетевые адреса. Сетевой уровень обеспечивает также прозрачную передачу пакетов на транспортный уровень.

Уровень 4, транспортный
Таранспортный уровень делит потоки информации на достаточно малые фрагменты (пакеты) для передачи их на сетевой уровень.

Уровень 5, сеансовый
Сеансовый уровень отвечает за организацию сеансов обмена данными между оконечными машинами. Протоколы сеансового уровня обычно являются составной частью функций трех верхних уровней модели.

Уровень 6, уровень представления
Уровень представления отвечает за возможность диалога между приложениями на разных машинах. Этот уровень обеспечивает преобразование данных (кодирование, компрессия и т.п.) прикладного уровня в поток информации для транспортного уровня.

Уровень 7, прикладной
Прикладной уровень отвечает за доступ приложений в сеть. Задачами этого уровня является перенос файлов, обмен почтовыми сообщениями и управление сетью.

3) Записать алгоритм решения системы линейных уравнений методом итераций.

Пусть дано:
y1 = a11*x1 + a12*x2 + a13*x3 + … + a1n*xn + b1
y2 = a21*x1 + a22*x2 + a23*x3 + … + a2n*xn + b2
…
yn = an1*x1 + an2*x2 + an3*x3 + … + ann*xn + bn

В общем виде система записывается так: $y_i = \sum_{j=1}^{n} a_{ij} * x_j + b_i$

необходимо определить применимость метода простых итераций к решению нашей системы линейных уравнений, для этого проверяем условие $\sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij}^2 < 1$, если условие выполняется, то метод применим, иначе нет, сам метод итераций выглядит следующим образом: $x_i^{(k+1)} = \sum_{j=1}^{n} a_{ij} * x_j^{(k)} + b_i$, приближения k, k+1, k+2, k+3…. вычисляем до тех пор пока не выполнится условие: $\sqrt{\sum_{i=1}^{n} (x_i^{(k)} - x_i^{(k+1)})^2} < \varepsilon$, где ε - точность.

Алгоритм:

1. НАЧАЛО
2. Sum := 0
3. цикл i от 1 до n
4. цикл j от 1 до n
5. Sum := Sum + a_{ij}
6. если Sum < 1 то переход на п.8 иначе переход на п.20
7. задаём начальное приближение x_i, i = 1…n
8. цикл i от 1 до n
9. $SumA_j$:= 0
10. цикл j от 1 до n
11. $SumA_j$:= $SumA_j + a_{ij}*x_{ij}$
12. конец цикла по j
13. $x_i$old = x_i
14. $x_i = SumA_j + b_i$
15. конец цикла по i
16. Er := 0
17. цикл i от 1 до n
18. Er := Er + $(x_i - x_i\text{old})^2$
19. если $\sqrt{Er} > \varepsilon$ то переходим на п.8 иначе переход на п.20
20. КОНЕЦ

Билет 6

1) Прямые методы решения вариационных задач.

1. Метод Ритца.

Прямые методы заключаются в нахождении искомой функции, доставляющей экстремум функционалу непосредственно его подбором. При этом не используется необходимое условие экстремума функционала и не составляется уравнение Эйлера. В методе Ритца предложено искать решение среди линейных комбинаций заранее заданных функций.

$$x = \sum_{i=0}^{n} a_i W_i(t) \qquad (**)$$

W_i - заранее известные заданные функции и не содержащие неизвестные коэффициенты.

a_i - неизвестные коэффициенты.

После подстановки (**) в функционал подынтегральная функция представляет собой набор известных функций аргумента t с неизвестными коэффициентами. Интеграл может быть взят, в результате чего получим некоторую функцию $\varphi(a_0, a_1, \ldots, a_n)$ и для неё надо найти экстремум. Поскольку необходимо определить экстремум n - переменных, то задача сводится от вариационной к конечномерной. Полученную задачу можно решить двумя методами:

1) Решив систему алгебраических уравнений.
2) Любым методом нелинейного программирования.

2. Метод Конторовича.

Имеет ту же основу что и метод Ритца, однако здесь допускается нелинейная комбинация искомых функций.

Приимущества:

Может быть достигнута лучшая аппроксимация экстремали при меньшем количестве параметров a_i, в то же время более сложен вопрос о подборе функций удовлетворяющих краевым условиям.

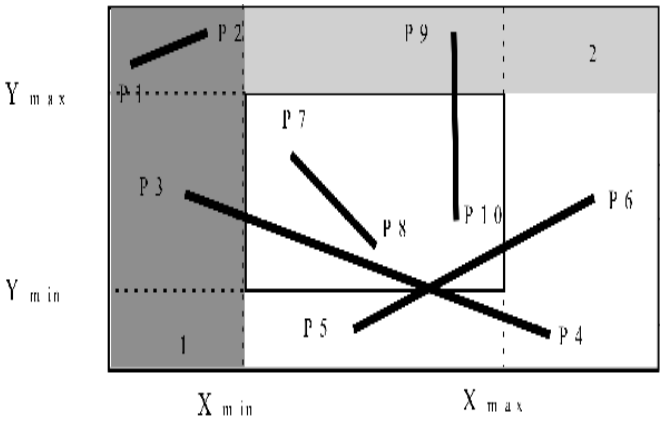
3. Конечноразностный метод Эйлера.

Интервал $[t_0, T]$ разбивается на n отрезков и ищутся значения функций в n-1 точке узлов разбиения. Искомая функция заменяется ломаной. Для каждой ломаной по методу прямоугольников, трапеций

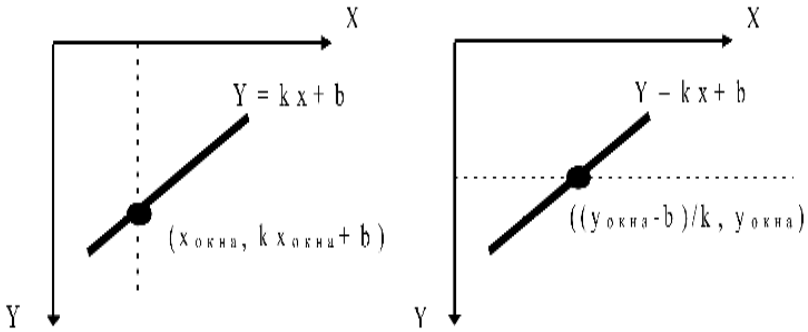
или Симпсона ищется значение функционала и находится та ломаная при которой значение функционала экстремально. С увеличением n точность аппроксимации увеличивается.

2) Окна в компьютерной графике. Алгоритмы преобразования координат при отсечении окном элементов изображения.

Функции окна:
1. удаление (удаляется то, что в окне)
2. отсечение (остается то, что в окне)
3. выделение



Рассмотрим отсечение графической модели областью 1 и 2:



Сначала:

$X_н$	$Y_н$	$X_к$	$Y_к$
x1	y1	x2	y2
x3	y3	x4	y4
x5	y5	x6	y
x7	y7	x8	y8
x9	y9	x10	y10

После отсечения областью 1:

$X_н$	$Y_н$	$X_к$	$Y_к$
x3’	y3’	x4	y4
x5	y5	x6	y
x7	y7	x8	y8
x9	y9	x10	y10

После отсечения областью 2:

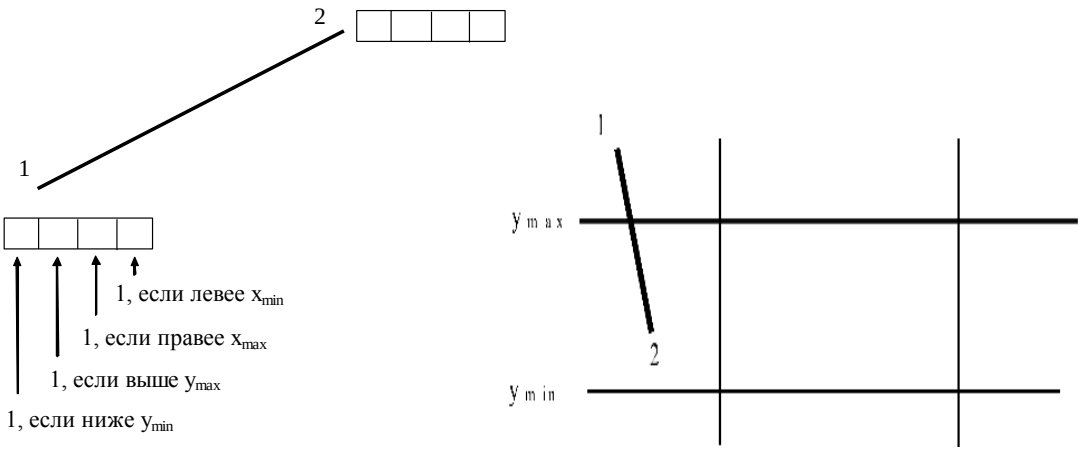
$X_н$	$Y_н$	$X_к$	$Y_к$
x3’	y3’	x4	y4

x5	y5	x6	y
x7	y7	x8	y8
x9’	y9’	x10	y10

и тд ....

На четвертом этапе в массиве будут находиться только те точки модели, которые находятся в окне.

Коэн и Сазерленд
Каждый отрезок прямой сопровождается информационным байтом (8 бит).



1: 1001
x
2: 0001

_____
0001

Если поразрядное логическое умножение дает итоговый результат 0, то отрезок находится внутри окна.

3) **Записать алгоритм определения объема кэш-памяти первого уровня.**

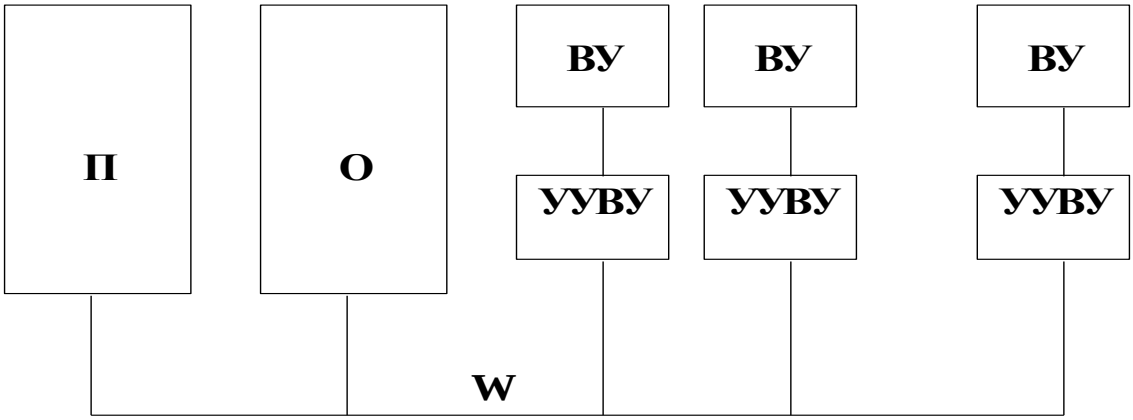
Билет 7

1) **Принцип структурной организации ЭВМ. Концепция единого интерфейса канала ввода-вывода.**

Чтобы реализовать заданный набор действий, необходима соответстствующая структура - определенный набор устройств, объединенных посредством связей (соединений) в одно целое. Структура определяет, как устроена ЭВМ, из каких физических частей она состоит и как эти части связаны друг с другом.

Концепция единого интерфейса и канала ввода—вывода

Наиболее естественным представляется способ соединения устройств, изображенный на рисунке. Процессор П, основная память О и внешние устройства *ВУ* подключаются к общей системе цепей — *единому интерфейсу W.* Для подключения внешних устройств используются *устройства управления УУВУ,* иначе называемые *контроллерами. УУВУ* — операционное устройство, назначение которого — реализация операций ввода—вывода в виде, соответствующем специфике функционирования внешнего устройства. *УУВУ* преобразует стандартную последовательность сигналов интерфейса в совокупность сигналов, обеспечивающую работу внешнего устройства. Для внешнего устройства каждого типа требуется своеобразное устройство управления. Объединение устройств *ЭВМ* посредством единого интерфейса порождает следующие принципы обмена информацией.



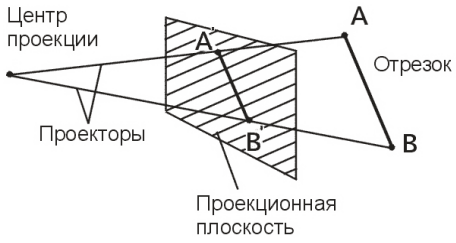
Единый интерфейс является высокоэффективным способом организации обмена информации в ЭВМ, которые комплектуются небольшим числом внешних устройств, работающих в режиме прямого доступа к памяти. Такая комплектация типична для мини и микро-ЭВМ. Поэтому единый интерфейс широко используется для ЭВМ этих классов.

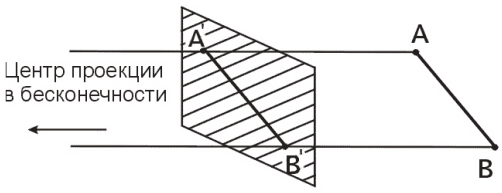
2) Проекции в трехмерной графике. Их математическое описание. Камера наблюдения.

Способ перехода от трехмерных объектов к их изображениям на плоскости будем называть проекцией.

Типы проекций:

Проекции строятся с помощью проецирующих лучей или проекторов, которые выходят из точки, которая называется центром проекции. Проекторы проходят через плоскость, которая называется проекционной или картинной плоскостью и затем проходят через каждую точку трехмерного объекта и образуют тем самым проекцию.
Если центр проекции находится на конечном расстоянии от проекционной плоскости, то проекция – центральная. Если же центр проекции удален на бесконечность, то проекция – параллельная.





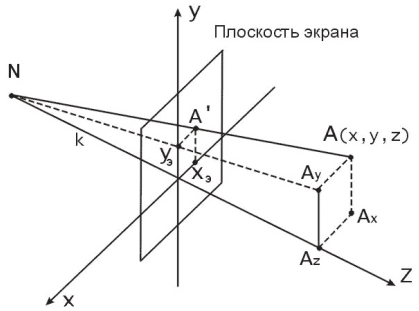
Центральная проекция. Параллельная проекция.

Точкой схода называется точка пересечения центральных проекций любой совокупности параллельный прямых, которые не параллельны проекционной плоскости. В зависимости от того,

сколько координатных осей пересекает проекционную плоскость различают одно-, двух- и трехточечные проекции.

Простейшей является параллельная прямоугольная проекция. В ней совместно изображаются виды сверху, спереди и сбоку. Эти проекции часто используются в черчении.

Рассмотрим более подробно центральную перспективную проекцию с математической точки зрения.



Точка A проецируется на экран как A'. Расстояние от наблюдателя до проекционной плоскости равно k. Необходимо определить координаты точки A' на экране. Обозначим их $x_э$ и $y_э$. Из подобия треугольников $A_y A_z N$ и $y_э ON$ находим, что

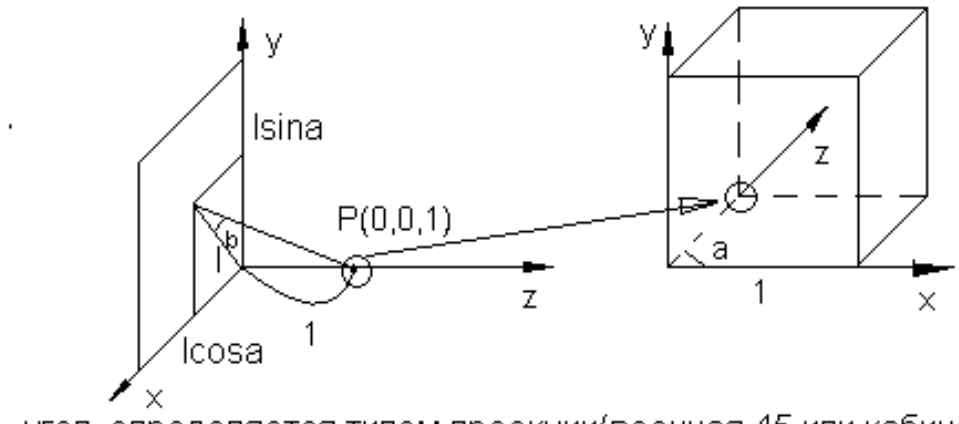
$$\frac{y}{z+k}=\frac{y_э}{k},\Rightarrow y_э=\frac{ky}{z+k} \qquad (1)$$

аналогично для x: $x_э=\frac{kx}{z+k}$.

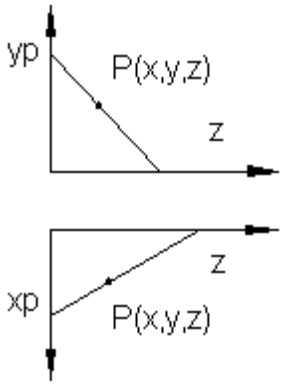
Напомним, что k -это расстояние, а наблюдатель находится в точке $N=(0,0,-k)$.

Если точку наблюдения поместить в начало координат, а проекционную плоскость на расстояние a, то формулы для $x_э$ и $y_э$ примут вид:

$$x_э=\frac{kx}{z},\qquad y_э=\frac{ky}{z} \qquad (2)$$



b - угол, определяется типом проекции(военная 45 или кабинетная 63,4)



Рассмотрим проекцию любой точки на плоскость x,y. Для этого рассмотрим вид в плоскости x=0 и y=0.

Запишем уравнение прямой

y=-lsina/1*z + yp => yp=y + lsina*z/1

x=-lcosa/1*z + xp => xp=x+lcosa*z/1

Для военной проекции l=1, для кабинетной l=0,5

3) Составить алгоритм поиска экстремума функции двух переменных методом случайного поиска.

$$\mathbf{F(x_1, x_2)=x^2_1 - x^2_2 + x_1x_2}$$

Метод случайного поиска заключается в прямой выборке из заданного интервала значений переменных с некоторой вероятностью

$$x_i = x_i + D * r_i$$

Где D – интервал поиска, а r – величина, равномерно распределенная в интервале от 0 до 1, или от -0,5 до 0,5.

После проверки точки на допустимость (попадание в интервал D) , вычисляется значение целевой функции в ней, если оно лучше предыдущего значения , то точка запоминается, иначе отбрасывается. Алгоритм работает указанное число итераций или пока не кончится машинное время.

1 Ввод функции f(x1,x2)

2 Ввод интервала D : a- левая граница, b – правая.

3 Ввод количества переменных N=2.

4 Ввод точности вычислений ε.

5 Ввод начальной точки $X^0(x_1^0, x_2^0)$

6 Ввод числа итераций E.

7 Для j от 1 до E //основной цикл, пока не кончатся итерации

Начало цикла

8 flag=0; min=1; // по умолчанию делаем минимум =1

9 для i от 1 до N

начало цикла

10. rand= Random((1-0.001)+0.001);

11. $x1_i = x_i + rand * fabs(a - b)$; // расчет новой точки X1

конец цикла

12. для i от 1 до N

начало цикла

13. Если $x1_i$ <a или $x1_i$ >b то flag=1; //проверка новой X1 на //принадлежность к интервалу

Конец цикла

14. Если flag=0 и f(X1)<min то X=X1 и min= f(X1)// если найденная точка X1 лучше предыдущей , то запоминаем её

Конец цикла

15. Вывод min и $X(x_1, x_2)$.

Билет 8

1) **Растровое представление геометрических объектов. Целочисленные алгоритмы для построения графических примитивов.**

Растровое - это представление объекта в том виде, в котором он будет выглядеть в реальности, то есть его изображение.

Целочисленные алгоритмы:

- Алгоритм Брезенхема для построения отрезков прямых.

Пусть на сетке растра задан отрезок прямой (x1,y1)-(x2,y2), тангенс угла наклона которого находится в диапазоне [0,1]. Для построения пэлов используют управляющую переменную d. На каждом шаге d пропорциональна разности между s и t. Для i-го шага, когда пэл P_{i-1} уже известен, требуется определить, какой из пэлов P_i или S_i должен быть выбран. Если s<t, то выбираем S_i, как наиболее близко расположенный пэл к реальному отрезку, в противном случае P_i. Начальное значение переменной d определяется как d=2*dy-dx, где dy = y2-y1 и dx = x2-x1. Для каждого значения $x_i = x_{i-1}+1$ проверяем знак управляющей переменной d:

а) если d>0, то выбираем пэл P_i, тогда $y_i=y_{i-1}+1$ и d=d+2*(dx-dy)

б) если d>0, то выбираем пэл S_i, тогда $y_i=y_{i-1}$ и d=d+2*dy

- Алгоритм Брезенхема для построения окружностей.

Аналогичным образом, но немного сложнее строится окружность. Классическая схема алгоритма Брезенхема рассматривается на случай обхода лишь дуги окружности в 45^0 от x = 0 до x=R/1.41.

Начальное значение управляющей переменной d определяется как d=3-2*R. Проверяем знак d:

а) d<0, $y_i=y_{i-1}$, d = d+4*x+6

б) d>0, $y_i=y_{i-1}$-1, d = d+4*(x-y)+10

Для построения полной окружности используется восьмисторонняя симметрия.

2) **Основные подходы к разработке ПО. Методы программирования и структура ПО.**

Программное обеспечение подразделяется на базовое, общесистемное и специализированное.

К основным методам программирования, ориентированным на получение надежных, пригодных для отладки, испытаний и сопровождения программ, можно отнести:

– Программирование на языках высокого уровня (на языках программирования, позволяющих разработчику абстрагироваться от особенностей используемой вычислительной машины, оперировать абстрактными типами данных).

– Структурное программирование (разработка программ модульной структуры с использованием ограниченного числа допустимых типов управляющих структур - последовательность, цикл, выбор).

– Программирование с защитой от ошибок (разработка программ, включающих дополнительные проверки входных и промежуточных данных на полноту, допустимость и правдоподобность получаемых значений).

– Программирование в стандартизированном стиле (разработка программ, оформление исходных текстов которых выполняется по единым для участников разработки правилам).

– Нисходящее проектирование (разработка программ путем поэтапного получения исходных модулей сначала верхних, затем нижних уровней иерархии).

3) **Исследовать на экстремум функцию:**
$F_0= x_1^2 - x_2^2 + 2x_3^2$,

при условии

$x_1 + x_2 + x_3 \geq 4$

Составляем функцию Лагранжа

$$L(\lambda, x) = f_0(x) + \sum_{j=1}^{m} \lambda_j f_j(x)$$

Достаточное условие:

- равенству нулю частных производных ф-ии Лагранжа по всем переменным.
- матрица вторых производных положительно определена.

$$L = x_1^2 - x_2^2 + 2x_3^2 + \lambda(x_1 + x_2 + x_3 - 4)$$

$$\frac{\partial L}{\partial x_1} = 0; \frac{\partial L}{\partial x_1} = 2x_1 + \lambda = 0$$

$$\frac{\partial L}{\partial x_2} = 0; \frac{\partial L}{\partial x_2} = -2x_2 + \lambda = 0$$

$$\frac{\partial L}{\partial x_3} = 0; \frac{\partial L}{\partial x_3} = 4x_3 + \lambda = 0$$

$$\frac{\partial L}{\partial \lambda} = x_1 + x_2 + x_6 - 4 = 0$$

Решая систему, находим $x_1 = 8; x_2 = -8; x_3 = 4; \lambda = -16$.

Матрица вторых производных :

$H = \begin{bmatrix} 2 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & 4 \end{bmatrix}$ - = -16 –> найденная точка является точкой максимума

Билет 9

1) **Основные подходы к разработке ПО САПР.**

Программное обеспечение подразделяется на базовое, общесистемное и специализированное.

К основным методам программирования, ориентированным на получение надежных, пригодных для отладки, испытаний и сопровождения программ, можно отнести:

* Программирование на языках высокого уровня (на языках программирования, позволяющих разработчику абстрагироваться от особенностей используемой вычислительной машины, оперировать абстрактными типами данных).
* Структурное программирование (разработка программ модульной структуры с использованием ограниченного числа допустимых типов управляющих структур - последовательность, цикл, выбор).
* Программирование с защитой от ошибок (разработка программ, включающих дополнительные проверки входных и промежуточных данных на полноту, допустимость и правдоподобность получаемых значений).
* Программирование в стандартизированном стиле (разработка программ, оформление исходных текстов которых выполняется по единым для участников разработки правилам).
* Нисходящее проектирование (разработка программ путем поэтапного получения исходных модулей сначала верхних, затем нижних уровней иерархии).

2) **Принципы построения и функционирования ЭВМ. Принцип программного управления.**

Все универсальные вычислительные машины, в том числе и персональные компьютеры, имеют структуру, показанную на рис, где обозначено:

АЛУ - арифметическо-логическое устройство;

УУ - устройство управления;
ВУ - внешние устройства;
ОЗУ - оперативное запоминающее устройство.
Линии со стрелками означают информационные и управляющие связи.

Общая структура универсальной ЭВМ.

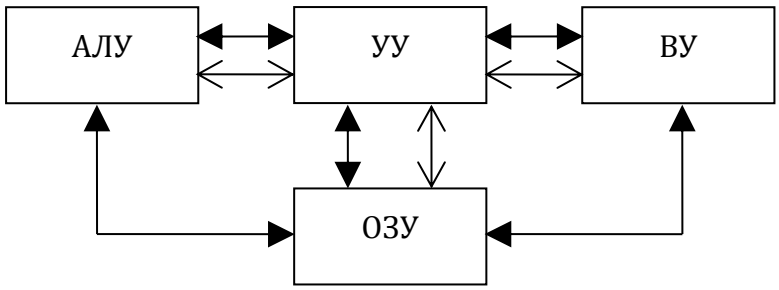


Рис.

Построение вычислительных машин основано на трех принципах:

- принцип цифрового представления данных;
- принцип адресности данных;
- принцип программного управления.

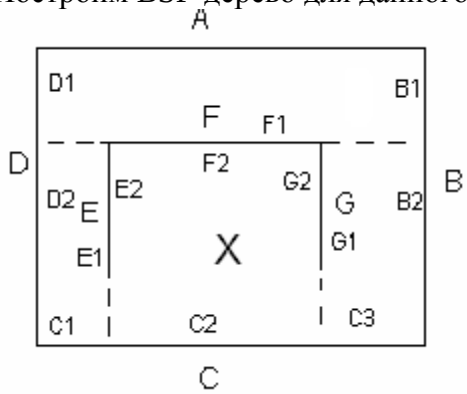
Принцип программного управления:

Принцип программы, хранимой в памяти компьютера, считается важнейшей идеей современной компьютерной архитектуры. Суть идеи заключается в том, что

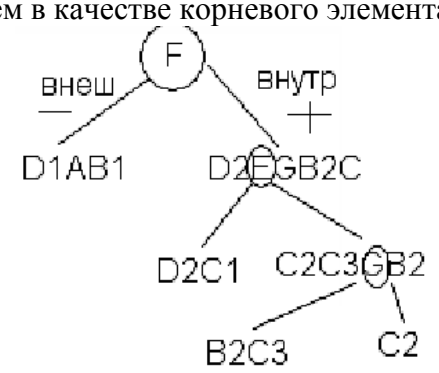
1) программа вычислений вводится в память ЭВМ и хранится в ней наравне с исходными числами;
2) команды, составляющие программу, представлены в числовом коде по форме ничем не отличающемся от чисел.

3) Показать пример построения BSP-дерева в алгоритмах удаления невидимых граней.

Построим BSP-дерево для данного помещения



Возьмем в качестве корневого элемента сторону F



Для показа BSP-дерева можно воспользоваться алгоритмом:
Например, мы находимся в точке X. Обращаемся в корневой элемент BSP-дерева. Если мы находимся внутри по отношению к F, то мы идем по дереву во внешнюю ветвь. Выписываем элементы ветви в отдельный список D1AB1. Далее поднимаемся в F и выписываем значение F. Идем в правую ветвь в узел E. Если мы находимся внутри по отношению к E, то идем во внешнюю ветвь. Выписываем D2C1EB2C3GC2.
Определение понятия внешней стороны:
Пусть требуется определить расположение полигона 2 и 3 по отношению к полигону 1. Для этого используют тестовую функцию:
T=Ax+By+Cz+D

Если в T полигона 1 подставить узловые точки полигона 2 и все значения будут положительны, то полигон 2 относится к правой ветви BSP-дерева, если отрицательны – то к левой. Если знаки меняются, то полигоны пересекаются.

Алгоритм визуализации BSP-дерева

Если камера находится в положительном подпространстве сцены по отношению к корневому элементу, то выводим отрицательную ветвь, далее корневой элемент и в конце выводим правую ветвь и это делается для каждого узла дерева рекурсивно.

Билет 10

1) **Организация диалога в САПР.**

Диалоговые языки служат средством оперативного взаимодействия проектировщика с ЭВМ, при котором происходит чередование запросов и ответов между человеком и ЭВМ в реальном масштабе времени.

По способам ввода команд различают более десяти типов представления языка диалога, среди которых наибольшее распространение получили языки типа «запрос — ответ» на основе:

1. директив пользователя;
2. выбора альтернативных возможностей;
3. заполнения пользователем форматов, представляемых машиной на экране дисплея.

В *директивных* языках основным форматом представления операторов является текстовая строка, а основным устройством ввода — алфавитно-цифровая клавиатура.

В языках, *основанных на процедуре выбора альтернативных возможностей,* конструирование предложения происходит путем указания предоставляемых на экране дисплея элементов словаря или выполнения действий с устройствами графического ввода.

Процедура трансляции предложений языка существенно упрощается, если синтаксический разбор начинается не по окончании ввода всего предложения, а по мере ввода его отдельных членов. Языки с такой организацией ввода получили название *диалоговых языков со сменными наборами команд* (СНК-языки).

Диалоговые языки, основанные на использовании графических изображений и устройств ввода графических данных, называют *диалоговыми графическими языками* (ДГ-языками).

2) **Видеоконтроллеры, их стандарты для ПЭВМ типа IBM PC.**

Первым стандартным видеоадаптером для IBM PC являлся MDA (Monochrome Display Adapter - монохромный адаптер дисплея). Он работал в текстовом режиме с разрешением 80x25, поддерживает пять атрибутов текста. Частота строчной развертки - 15 кГц. Интерфейс с монитором - цифровой. Объем видеопамяти 4 кБ.

В восьмидесятые годы конкуренцию MDA составлял HGC (Hercules Graphics Card - графическая карта Hercules) - тот же самый MDA плюс монохромный графический режим 720x348.

Первым видеоадаптером для IBM PC, поддерживающим цветные графические режимы являлся CGA (Color Graphics Adapter - цветной графический адаптер). Он работает либо в текстовом режиме с разрешениями 40x25 и 80x25, либо в графическом с разрешениями 320x200 или 640x200. Частота строчной развертки - 15 кГц. Интерфейс с монитором - цифровой.

EGA (Enhanced Graphics Adapter - улучшенный графический адаптер) - дальнейшее развитие CGA. Добавлено разрешение 640x350. Частоты строчной развертки - 15 и 18 кГц. Интерфейс с монитором - цифровой.

MCGA (Multicolor Graphics Adapter - многоцветный графический адаптер) - введен фирмой IBM в ранних моделях PS/2. Добавлено разрешение 640x400 (текст). Частота строчной развертки - 31 кГц. Интерфейс с монитором - аналогово-цифровой.

VGA (Video Graphics Array - множество, или массив, визуальной графики) - расширение MCGA, совместимое с EGA, введен фирмой IBM в средних моделях PS/2.

IBM 8514/a - специализированный адаптер для работы с высокими разрешениями (640x480x256 и 1024x768x256), с элементами графического ускорителя.

IBM XGA - следующий специализированный адаптер IBM. Расширено цветовое пространство (режим 640x480x64k), добавлен текстовый режим 132x25 (1056x400). Как и IBM 8514/a не получил широкого распространения.

SVGA (Super VGA - "сверх"-VGA) - расширение VGA с добавлением более высоких разрешений.

3) Текстуры в машинной графике. Фильтрация текстур.

Задача текстурирования формулируется таким образом:
есть грань - согласно предположениям, треугольная - с наложенной на нее текстурой. То есть каждая точка грани окрашена цветом соответствующей ей точки в текстуре. Текстура накладывается линейным образом. Есть точка экрана с координатами на экране (sx,sy), принадлежащая проекции грани. Требуется найти ее цвет, то есть цвет соответствующей этой точке экрана точки текстуры. А для этого надо найти координаты текстуры для этой точки - точнее, для той точки, проекцией которой на экран является наша (sx,sy).

Аффинное текстурирование:
Пусть есть некоторая точка плоскости экрана (sx,sy), соответствующая точки пространства (x,y,z), тогда (u,v) – соответствующая точка текстуры.
Этот метод текстурирования основан на приближении u, v линейными функциями. Итак, пусть u - линейная функция, u = k1*sx+k2*sy+k3. Можно посчитать k1, k2, k3 исходя из того, что хотя бы в вершинах грани u должно совпадать с точным значением - это даст нам три уравнения, из которых быстро и просто находятся эти коэффициенты, и потом считать u по этой формуле.

Билинейная фильтрация текстур:
При билинейной фильтрации цвет всего-навсего линейно интерполируется между узлами сетки замеров.
Используя линейную интерполяцию вычислим интенсивность в некоторой точке A:

$$Ca = C00(1-(xt-[xt]))+C10(([xt]+1)-xt)$$
$$Cb = C01(1-(xt-[xt]))+C11(([xt]+1)-xt)$$
$$Ct = Ca(1-(yt-([yt]-1)))+Cb(([yt]+1)-yt)$$

Мипмэппинг:
При использовании мипмэппинга текстурирование осуществляется следующим образом:

- начиная с базового уровня определяется количество текселей на один пиксель
- если выясняется, что на 1 пиксель приходится некратное 4 число текселей, используют линейную фильтрацию текстур (выборка текселей из 2-х соседних уровней текстур)

Билет 11

1) Этапы решения задачи с использованием ЭВМ.

Обычно выделяют следующие этапы решения задачи:

1) Постановка задачи – определяются цели и условия решения задачи, раскрывается ее содержание, выявляются факторы, оказывающие влияние на ход вычислений или конечный результат.
2) Формализация – по результатам выяснения сущности задачи определяется объект и специфика исходных данных, вводится система условных обозначений, устанавливается

принадлежность поставленной задачи к одному из известных классу задач и выбирается математический аппарат описания.

3) Выбор или разработка метода решения. Построение на предыдущем этапе мат модели приводит к необходимости поиска ее решения, т е устанавливается зависимость результатов от исходных данных, подбираются методы решения задачи, пригодные для реализации на ЭВМ.
4) Составления алгоритма – часто процесс решения задачи не удается получить в виде явной формулы. В этом случае метод решения задачи преобразовывается в последовательность действий ЭВМ, называемой алгоритмом.
5) Составление программы – составленный алгоритм записывается на языке программирования, после чего программа компилируется в коды ЭВМ. Результат – рабочая программа.
6) Отладка программы – на любом из предшествующих этапов могли быть допущены ошибки, которые делают код программы неработоспособным. Отладка состоит в выявлении и устранении грубых ошибок.
7) Тестирование – тесно связано с отладкой. Однако предназначено для выявления грубых ошибок, которые не нарушают работоспособность программы, но не дают ей возможности выдавать правильный результат.

2) Классификация центральных процессоров Intel и соответствующих локальных и системных шин ПЭВМ типа IBM PC.

Классификация:

- 4-битные процессоры (4004: первый процессор, реализованный в одной микросхеме, 4040) 1971-1972 г.;
- 8-битные процессоры (8008, 8080, 8085) 1972-1976 г.;
- 16-битные процессоры: Происхождение x86 (8086, 8088, 80186, 80188, 80286) 1978-1982 г.;
- 32-битные процессоры: Линия 80386 (80386DX, 80386SX, 80376, 80386SL, 80386EX) 1985-1990 г.;
- 32-битные процессоры: Линия 80486 (80486DX, 80486SX, 80486GX, 80486DX2, 80486SL, 80486DX4) 1989-1994 г.;
- 32-битные процессоры: Pentium I (Pentium («Классический»), Pentium MMX) 1993, 1997;
- 32-битные процессоры: микроархитектура P6/Pentium M (Pentium Pro, Pentium II, Celeron (Pentium II-based), Pentium III, Pentium II и III Xeon, Celeron (Pentium III базирующийся на ядре Coppermine), Celeron (Pentium III на ядре Tualatin), Pentium M, Celeron M, Intel Core, Pentium Dual-Core, Dual-Core Xeon LV) 1995-1999 г.;
- 32-битные процессоры: микроархитектура NetBurst (Pentium 4, Xeon) 2000-2001 г.;
- 64-битные процессоры: IA-64(Itanium, Itanium 2, Itanium 2 серия 9000, Itanium 2 серия 9100) 2001 г.;
- 64-битные процессоры: EM64T — Микроархитектура NetBurst (Pentium 4F, D0 и более поздние степпинги, Pentium D, Pentium Extreme Edition, Xeon) 2005 г.;
- 64-битные процессоры: EM64T — Микроархитектура Intel Core (Xeon, Intel Core 2, Pentium Dual Core, Celeron Dual Core, Celeron (микроархитектура Core), Celeron M (микроархитектура Core)) 2007-2010 г.;
- Intel Atom (32-битные процессоры: IA-32, 64-битные процессоры: EM64T) 2008 г.;
- 64-битные процессоры: EM64T — Микроархитектура Nehalem (Intel Celeron, Intel Pentium, Intel Core i7, Intel Core i5, Intel Core i3, Intel Core i7 Extreme Edition, Intel Xeon (UP/DP), Intel Xeon MP) 2011 г.;
- 64-битные процессоры: EM64T — Микроархитектура Sandy Bridge (Intel Celeron, Intel Pentium, Intel Core i3, Intel Core i5, Intel Core i7, Intel Core i7 Extreme Edition, Intel Xeon E3);
- 64-битные процессоры: EM64T — Микроархитектура Ivy Bridge (Intel Core i3, Intel Core i5, Intel Core i7) 2012 г..

В компьютерах PC/AT стала применяться 16-разрядная системная шина ISA (Industry Standart Architecture). С появлением процессоров 80386, а затем и 80486 системная шина ISA стала тормозом на пути увеличения общей производительности системы. Новой шиной стала EISA (Extended Industry Standart Architecture). В пакетном режиме она обеспечивала теоретическую скорость 33 Мб/с. Однако, данная шина не получила большого распространения и использовалась преимущественно в мощных файл-серверах и рабочих станциях. Шина PCI является надстройкой над локальной шиной процессора, т.к. связана с ней логически и электрически с помощью специального контроллера - «моста» (bridge), благодаря чему одновременно к шине может быть подключено до 10 устройств.

3) Реалистическая графика. Обратная трассировка луча. Полная модель освещения Уиттеда.

С развитием вычислительной техники и ее возможностей все больший интерес вызывает задача построения по заданной модели сцены изображения, максимально приближенного к реальному. Построенное таким образом изображение должно восприниматься человеком как настоящее, реалистическое.

Для избавления от излишних вычислений используется обратная трассировка, в которой вычисляются интенсивности только лучей, попавших в глаз наблюдателя.

Идея обратной трассировки лучей:
Для определения цвета пиксела экрана через него из камеры проводится луч, ищется его ближайшее пересечение со сценой и определяется освещенность точки пересечения. Эта освещенность складывается из отраженной и преломленной энергий, непосредственно полученных от источников света, а также отраженной и преломленной энергий, идущих от других объектов сцены. После определения освещенности этой точки учитывается ослабление света при прохождении через прозрачный материал и в результате получается цвет точки экрана.

Для определения освещенности, привносимой непосредственным освещением, из точки пересечения выпускаются лучи ко всем источникам света и определяется вклад всех источников, которые не заслонены другими объектами сцены. Для определения отраженной и преломленной освещенности из точки выпускаются отраженный и преломленный лучи и определяется привносимая ими освещенность.

Простейшей моделью освещенности является модель Уиттеда, согласно которой световая энергия, покидающая точку в направлении вектора, определяется формулой

$$I(\lambda) = k_a I_a(\lambda) + k_d C(\lambda) \sum_j I_j(\lambda)(n, l_j) + k_r I_r(\lambda) e^{-\beta_r d_r} + k_s \sum_j I_j(\lambda)(n, h_j)^p + k_t I_t(\lambda) e^{-\beta_t d_t}$$

Где:

- k_a, k_d,k_s- веса фонового, диффузного и зеркального освещения;
- k_r, k_t - веса отраженного и преломленного лучей;
- C(l) - цвет объекта в точке P;
- n - вектор нормали в точке P;
- l_j - направление на j-е источник света;
- I_j(l) - цвет j-го источника света;
- b_r , b_t - коэффициенты ослабления для сред, содержащих отраженный и преломленный лучи;
- d_r , d_t - расстояния от точки P до ближайших точек пересечения отраженного и преломленного лучей с объектами сцены;
- h_j - вектор, задаваемый формулой
- p - коэффициент Фонга.

Билет 12

1) **Язык SQL. Операторы языка для описания и манипулирования данными.**

SQL (*Structured Query Language* — «язык структурированных запросов») — универсальный компьютерный язык, применяемый для создания, модификации и управления данными в реляционных базах данных.

Операторы:

- операторы определения данных (*Data Definition Language, DDL*)

  CREATE создает объект БД (саму базу, таблицу, представление, пользователя и т. д.)
  ALTER изменяет объект
  DROP удаляет объект

- операторы манипуляции данными (*Data Manipulation Language, DML*)

  SELECT считывает данные, удовлетворяющие заданным условиям
  INSERT добавляет новые данные
  UPDATE изменяет существующие данные
  DELETE удаляет данные

- операторы определения доступа к данным (*Data Control Language, DCL*)

  GRANT предоставляет пользователю (группе) разрешения на определенные операции с объектом
  REVOKE отзывает ранее выданные разрешения
  DENY задает запрет, имеющий приоритет над разрешением

- операторы управления транзакциями (*Transaction Control Language, TCL*)

  COMMIT применяет транзакцию.
  ROLLBACK откатывает все изменения, сделанные в контексте текущей транзакции.
SAVEPOINT делит транзакцию на более мелкие участки.

2) **Цвет в машинной графике. Аппроксимация полутонами.**

В общем случае цвет представляет собой набор чисел, координат в некоторой цветовой системе.

Стандартные способы хранения и обработки цвета в компьютере обусловлены свойствами человеческого зрения. Наиболее распространены системы RGB для дисплеев и CMYK для работы в типографском деле.

Аппроксимация полутонами:

- Метод нарастающей последовательности

Для создания несуществующего оттенка серого цвета на мониторе используется этот метод.

5 градаций серого.
N*N + 1, где N- размерность матрицы возбуждения.

$$\begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix}$$

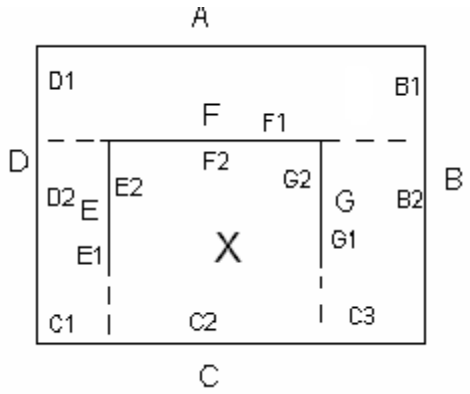
Для цвета укрупненные пэлы как бы разбавляются двумя основными цветами в третий.
0 1 2 3 - 4 градации зеленого.
1- красный
2- зеленый.

0	0	1	0	1	0	1	1	1	1	2	1	2	1	2	2	2	2
0	0	0	0	0	1	0	1	1	1	1	1	1	2	1	2	2	2

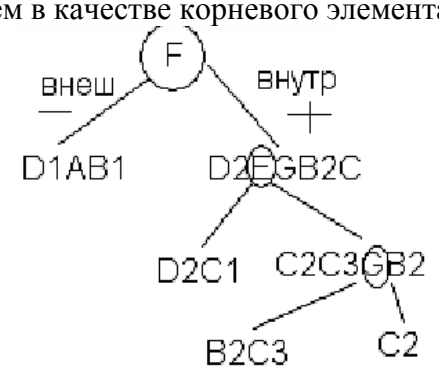
- Метод упорядоченного возбуждения

Пусть требуется вывести растровое изображение n*m на черно-белое устройство. Для каждой точки изображения будем сравнивать соотв. Значения из матрицы возбуждения и яркость.
Если I(x,y)>D[x mod n, y mod m], то выводим по координате (x,y) 1, иначе выводим 0

3) Показать пример построения BSP-дерева в алгоритмах удаления невидимых граней.



Построим BSP-дерево для данного помещения
Возьмем в качестве корневого элемента сторону F



Для показа BSP-дерева можно воспользоваться алгоритмом:
Например, мы находимся в точке X. Обращаемся в корневой элемент BSP-дерева. Если мы находимся внутри по отношению к F, то мы идем по дереву во внешнюю ветвь. Выписываем элементы ветви в отдельный список D1AB1. Далее поднимаемся в F и выписываем значение F. Идем в правую ветвь в узел E. Если мы находимся внутри по отношению к E, то идем во внешнюю ветвь. Выписываем D2C1EB2C3GC2.
Определение понятия внешней стороны:
Пусть требуется определить расположение полигона 2 и 3 по отношению к полигону 1. Для этого используют тестовую функцию:
T=Ax+By+Cz+D
Если в T полигона 1 подставить узловые точки полигона 2 и все значения будут положительны, то полигон 2 относится к правой ветви BSP-дерева, если отрицательны – то к левой. Если знаки меняются, то полигоны пересекаются.
Алгоритм визуализации BSP-дерева
Если камера находится в положительном подпространстве сцены по отношению к корневому элементу, то выводим отрицательную ветвь, далее корневой элемент и в конце выводим правую ветвь и это делается для каждого узла дерева рекурсивно.

Билет 13

1) Структурное программирование при разработке программ.

Структурное программирование — методология разработки программного обеспечения, в основе которой лежит представление программы в виде иерархической структуры блоков. Предложена в 70-х годах XX века Э. Дейкстрой, разработана и дополнена Н. Виртом.

В соответствии с данной методологией:

a) Любая программа представляет собой структуру, построенную из трёх типов базовых конструкций:
   - **последовательное исполнение** — однократное выполнение операций в том порядке, в котором они записаны в тексте программы;
   - **ветвление** — однократное выполнение одной из двух или более операций, в зависимости от выполнения некоторого заданного условия;
   - **цикл** — многократное исполнение одной и той же операции до тех пор, пока выполняется некоторое заданное условие (условие продолжения цикла).
b) В программе базовые конструкции могут быть вложены друг в друга произвольным образом, но никаких других средств управления последовательностью выполнения операций не предусматривается.
c) Повторяющиеся фрагменты программы могут оформляться в виде т. н. подпрограмм (процедур или функций).

Разработка программы ведётся пошагово, методом «сверху вниз».

2) Понятие критерия оптимального проектирования и его связь с варьируемыми переменными через уравнения математической модели. Постановка задачи оптимального проектирования.

Задачей проектирования новых объектов на современном этапе является получение не просто работоспособных, но и оптимальных с точки зрения заданного критерия объекта.

Для постановки и решения задачи оптимального проектирования необходимо два фактора:
1. Должны быть варьируемые переменные, изменяя которые проектировщик получает различные проектные решения
2. Критерий, по которому сравниваются проектные решения

Для связи варьируемых переменных с критерием используется математическая модель (ММ).

Математическая модель – это система булевских, алгебраических, дифференциальных, интегральных уравнений, связывающих варьируемые переменные с критерием.

В качестве критериев могут использоваться:
1. Технологические переменные: производительность, концентрация на выходе, температура на выходе и т.д.
2. Конструкционные характеристики: объем, длина и др. габаритные размеры.
3. Технико-экономические показатели: себестоимость, прибыль, технологическая себестоимость и др.

Постановка задачи:

Найти значения варьируемых переменных X_1^*, X_2^*, ..., X_n^*, при которых критерий достигает экстремального значения при выполнении ограничений, наложенные на входные и выходные координаты, и выполнении связей, заданных в виде уравнений математической модели.

3) Представить алгоритм определения быстродействия НГМД в режиме записи данных.

```
program FDD_Speed;
uses
  Crt,Dos;
```

```
var
  i,Free,T1,T2      :LongInt;
  F                 :Text;
  Hour,
  Min1,Min2,
  Sec1,Sec2,
  Sec100_1,Sec100_2 :Word;

begin
  ClrScr;
  Free:=DiskSize(1);
  WriteLn('Свободно на дискете: ',Free,' байт');

  Assign(F,'a:\a.$$$'); ReWrite(F);

  GetTime(Hour,Min1,Sec1,Sec100_1);
  T1:=Min1*60+Sec1;
  for i:=1 to Free do
    Write(F,'a');
  GetTime(Hour,Min2,Sec2,Sec100_2);
  T2:=Min2*60+Sec2;

  WriteLn('Время    записи : ',(T2-T1):2:2,' сек');
  WriteLn('Скорость записи : ',(Free/(T2-T1)):4:1,' байт/сек');
  Close(F);
end.
```

Билет 14

1) **Базы данных. Этапы проектирования.**

База данных— представленная в объективной форме совокупность самостоятельных материалов (статей, расчётов, нормативных актов, судебных решений и иных подобных материалов), систематизированных таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины (ЭВМ).

Этапы проектирования базы данных:

1. Концептуальное проектирование — сбор, анализ и редактирование требований к данным. Для этого осуществляются следующие мероприятия:
   - обследование предметной области, изучение ее информационной структуры
   - выявление всех фрагментов, каждый из которых характеризуется пользовательским представлением, информационными объектами и связями между ними, процессами над информационными объектами
   - моделирование и интеграция всех представлений

   По окончании данного этапа получаем концептуальную модель, инвариантную к структуре базы данных. Часто она представляется в виде модели «сущность-связь».
2. Логическое проектирование — преобразование требований к данным в структуры данных. На выходе получаем СУБД-ориентированную структуру базы данных и спецификации прикладных программ. На этом этапе часто моделируют базы данных применительно к различным СУБД и проводят сравнительный анализ моделей.
3. Физическое проектирование — определение особенностей хранения данных, методов доступа и т. д.

2) **Оптимизационные задачи в автоматизированных системах технологической подготовки производства.**

Примеры оптимизационных задач при ТПП:

- Найти материал детали, обеспечивающий минимум ее стоимости при выполнении заданных требований.
- Найти форму и метод изготовления заготовки, обеспечивающие минимум потерь материала.
- Определить последовательность технологических переходов, обеспечивающую минимальное время изготовления партии деталей.
- Выбрать оборудование, обеспечивающее:
  а) минимальную стоимость при удовлетворении требований техпроцесса;
  б) минимальные приведенные затраты на выполнение технологического контроля;
  в) минимальный период окупаемости оборудования.

Оптимизационные задачи, решаемые при автоматизации метода нового планирования:
Автоматизация этого метода наиболее трудоемка, т.к. при его использовании осуществляется проектирование и документирование ТП на основе введенных данных.

1. **Выбор вида заготовки и методов ее изготовления** Виды заготовок: отливки; прокат; поковки; штамповки; сварные заготовки. В качестве критериев оптимизации выбора заготовок используют:
   – себестоимость изготовления заготовки Сз → min;
   – себестоимость механической обработки заготовки для получения детали См → min;
   – стоимость отходов металла Со → min.
2. **Выбор технологических баз**
3. **Проектирование технологического маршрута** Данная задача - главная и наиболее трудная. В методе нового планирования используют различные диалоговые подсистемы формирования технологического маршрута.
4. **Проектирование технологических операций** Каждая технологическая операция, выбранная на этапе проектирования технологического маршрута, проектируется в виде последовательности переходов. Выбор наилучшего варианта осуществляется по критериям: себестоимость операции; время выполнения операции и другим.
5. **Выбор основного оборудования** Оборудование для выполнения операций выбирается в зависимости от намеченного состава операций, габаритов и конфигурации детали, требуемой точности обработки, программы выпуска деталей.
6. **Выбор инструмента** Выбор режущего инструмента осуществляется для каждого технологического перехода.
7. **Оптимизация проектирования сборочных процессов** Сборочные работы являются многовариантными как по возможному составу и последовательности операций техпроцесса, так и по составу применяемой оснастки, оборудования, инструмента. В качестве критериев оптимизации используются:
   – трудоемкость процесса сборки;
   – технологическая себестоимость;
   – цикл сборки (время);
   – затраты на сборочную оснастку.

3) Таблицы истинности, совершенные нормальные формы представления булевых функций.

Таблица истинности — это таблица, описывающая логическую функцию.

Под «логической функцией» в данном случае понимается функция, у которой значения переменных (параметров функции) и значение самой функции выражают логическую истинность. Например, в двузначной логике они могут принимать значения «истина» либо «ложь» (1 или 0).

Совершенная дизъюнктивная нормальная фо́рма (СДНФ) — это такая ДНФ, которая удовлетворяет трём условиям:

- в ней нет одинаковых элементарных конъюнкций
- в каждой конъюнкции нет одинаковых пропозициональных букв

- каждая элементарная конъюнкция содержит каждую пропозициональную букву из входящих в данную ДНФ пропозициональных букв, причём в одинаковом порядке.

Для любой функции алгебры логики существует своя СДНФ, причём единственная.

Совершенная конъюнктивная нормальная форма (СКНФ) — это такая КНФ, которая удовлетворяет трём условиям:

- в ней нет одинаковых элементарных дизъюнкций
- в каждой дизъюнкции нет одинаковых пропозициональных переменных
- каждая элементарная дизъюнкция содержит каждую пропозициональную букву из входящих в данную КНФ пропозициональных букв.

Билет 15

1) Методы безусловной оптимизации нелинейного программирования.

Метод покоординатного спуска

Рассмотрим вначале метод покоординатного спуска Гаусса-Зейделя на примере функции двух переменных, линии равного уровня которой изображены на рис.2.6. Из некоторой начальной точки $x^0=(x_1^0, x_2^0)$ производится поиск минимума вдоль направления оси x_1 с получением точки X^0. В этой точке касательная к линии равного уровня параллельна оси x_1. Затем из точки X^0 производится поиск минимума вдоль направления оси x_2 с получением точки x^1. Следующие итерации выполняются аналогично. Для минимизации функции $f_0(x)$ вдоль осевых направлений может быть использован любой из методов одномерной минимизации.

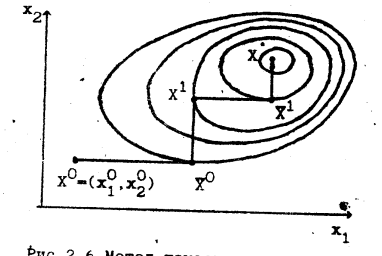


Рис. 2.6 Метод покоординатного спуска Гаусса-Зейделя

54

Критерием окончания поиска является выполнение условия:

$$\left\|x^{k+1}-x^{k}\right\| \le \varepsilon, \quad \text{где} \quad \left\|x^{k+1}-x^{k}\right\| = \sqrt{\sum_{i=1}^{n}(x_i^{k+1}-x_i^{k})^2}$$

Метод Пауэлла

После получения точки $\bar{X}^0$ локальным поиском вдоль координатных осей выполняется поиск вдоль направления, соединяющего точки X^0 и $\bar{X}^0$ (рис. 2.7) с получением точки X^1, т.е. $X^1=X^0+h_0(\bar{X}^0-X^0)$, где h_0 вычисляется из условия того, что точка X^1 является точкой минимума функции f_0 вдоль направления $S^0=(\bar{x}_1^0-x_1^0, \bar{x}_2^0-\bar{x}_2^0)$, т.е. $h_0=\text{argmin}(f_0(X°+hS°)$

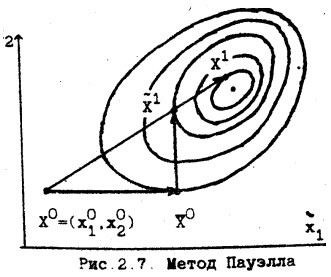


Рис.2.7. Метод Пауэлла

Значение h_0 отыскивается любым из методов одномерной минимизации. Для локализации отрезка $[h', h'']$, содержащего минимум по h, необходимо использование процедуры локализации, описанной выше. При этом очевидно, что в точке X^0 $h_0=0$, в точке $\bar{X}^0$ $h_0=1$.

Следующие итерации выполняются аналогично.

Симплексный метод

Алгоритм решения задачи симплексным методом может быть представлен в виде следующей последовательности шагов.

1. Задаются n+1 точка $X^1, X^2, ..., X^{n+1}$, являющиеся вершинами начального симплекса.

2. Вычисляются величины $f_1 = f_0(X^1), \ldots, f_{n+1} = f_0(X^{n+1})$.

3. Определяются вершины h, g, l, для которых f_h - наибольшее значение среди всех f_i, i=l,… , n+l; f_g - следующее за наибольшим; f_l - наименьшее значение.

4. Определяется центр тяжести X^C всех точек, кроме X^h

$$X^C = \frac{1}{n} \sum_{\substack{i=1 \\ i \neq h}}^{n} X^i,$$

и вычисляется значение целевой функции f_0 в точке $X^C - f_C$.

5. Выполняется операция отражения точки X^h относительно точки X^C с коэффициентом отражения α и получением точки X^0 (рис. 2.8). При этом $X^0 - X^C = \alpha (X^C - X^h)$, откуда $X^0 = (1+\alpha) X^C - \alpha X^h$. Вычисляется значение целевой функции f_0, в точке $X^0 - f_0$.

6. Значение f_0 сравнивается со значениями f_1 и f_g. Возможны три случая:

6.1. Если $f_0 < f_1$, то направление из точки X^C в точку X^0 является наиболее предпочтительным для перемещения. В этом случае выполняется операция растяжения симплекса с коэффициентом растяжения β и получением точки X^r. При этом $X^r - X^C = \beta (X^0 - X^C)$,
откуда $X^r = (1-\beta) X^C - \beta X^0$.

Далее вычисляется значение целевой функции f_0 в точке $X^r - f_r$ и осуществляется его сравнение со значением f_1. Возможны два случая:

6.1.1. Если $f_r < f_1$, то точка X^h перемещается в точку X^r, образуя новый симплекс, текущая итерация заканчивается и выполняется переход к шагу 10 для проверки критерия останова.

6.1.2. Если $f_r \geq f_1$, то точка X^r отбрасывается, т.к. перемещение было выполнено слишком далеко. В этом случае точка X^h перемещается в точку X^0, образуя новый симплекс, текущая итерация заканчивается и выполняется переход к шагу 10 для проверки критерия останова.

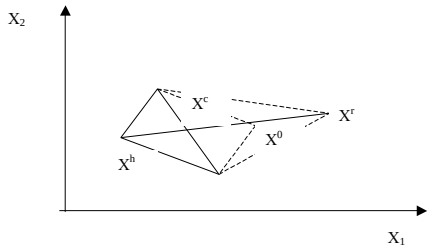


Рис.2.8. Операция отражения и растяжения симплекса

6.2. Если $f_1 < f_0 \leq f_g$, то X^0 является не самой плохой точкой. В этом случае точка X^h перемещается в точку X^0 и выполняется переход к шагу 10.

6.3. Если $f_0 > f_g$, то выполняется переход к шагу 7.

7. Выполнение операции сжатия симплекса. Возможны два варианта сжатия в зависимости от

значений f_0 и f_h.

7.1. Если $f_0 < f_h$, то результатом операции сжатия является точка X^S, определяемая из условия $X^S - X^C = \gamma (X^0 - X^C)$ или $X^S = \gamma X^0 + (1-\gamma) X^C$

В этом случае точка X^S будет лежать между X^C и X^0.

7.2. Если $f_0 \geq f_h$, то результатом операции сжатия является точка X^S, определяемая из условия

$$X^S - X^C = \gamma (X^h - X^C) \text{ или } X^S = \gamma X^h + (1-\gamma) X^C.$$

В этом случае точка X^S будет лежать между X^h и X^C (рис.2.9б). Далее вычисляется значение целевой функции в точке $X^S - f_S$.

8. Сравниваются значения f_h и f_S.

8.1. Если $f_S < f_h$, то точка X^h перемещается в точку X^S, образуя новый симплекс, текущая итерация заканчивается и выполняется переход к шагу 10.

8.2. Если $f_S \geq f_h$, то точки со значением функции, меньшим, чем f_h, найти не удалось. В этом случае выполняется переход к шагу 9.

9. Выполнение операции уменьшения размеров симплекса. Размерность симплекса уменьшается относительно точки X^l путем уменьшения в 2 раза расстояний от точки X^l до всех остальных вершин симплекса (рис.2.10). При этом

$$X^i = X^i + \frac{1}{2}(X^i - X^1),\ i \neq 1.$$

Затем вычисляются значения $f_i = f_0(X^i)$, i=l,..., n+l, и выполняется переход к шагу 10.

10. Проверка критерия останова. Процесс поиска прекращается, если по завершении очередной k-й итерации выполняется условие

$$\sigma < \varepsilon, \text{ где } \sigma^2 = \sum_{i=1}^{n+1} (f_i - f)^2 /(n+1) \text{ и } f = \sum_{i=1}^{n+1} f_i /(n+1).$$

Если условие $\sigma < \varepsilon$ не выполняется, то осуществляется переход к шагу 3.

В качестве значений коэффициентов отражения, растяжения и сжатия рекомендуется использовать: $\alpha = 1,\ \delta = 2,\ \gamma = 0{,}5$. Для формирования начального симплекса задается точка X^1, остальные точки вычисляются по формулам:

$X^2 = X^1 + kE^1,\ X^3 = X^2 + kE^2, \ldots, X^{n+1} = X^n + kE^n$, где

$E^j = (e_1, e_2, \ldots, e_j, \ldots, e_n)$ – вектор, у которого $e_j = 1$, а остальные элементы равны 0, j=1,…, n, k - произвольная длина шага.

Метод наискорейшего спуска

Рассмотрим использование метода наискорейшего спуска для минимизации функции двух переменных $f_0(x_1, x_2)$. Пусть в качестве начального приближения выбрана некоторая точка $X^0 = (x_1^0, x_2^0)$. Каждая итерация при поиске минимума методом наискорейшего спуска состоит из двух этапов. На первом этапе в точке X^0 вычисляются значения частных производных по переменным x_1, x_2, которые определяют направление градиента функции $f_0(X)$ в этой точке. На втором этапе определяется следующая точка X^1 как $X^1 = X^0 + h_0 S^0$, где $S^0 = -\nabla f_0(X^0) = \left[-\frac{\partial f_0(X^0)}{\partial x_1} - \frac{\partial f_0(X^0)}{\partial X_2} \right]$ - антиградиент функции f_0 в точке X^0, h_0 - неизвестное значение коэффициента h.

Значение h_0 вычисляется из условия того, что точка X^1 является точкой минимума функции $f_0(X)$ вдоль направления антиградиента, т.е.

$h_0 = \arg\min f_0(X^0 + hS^0)$.

Для произвольных k-й итерации и функции n переменных следующее приближение X^{k+1} к точке минимума X вычисляется как

$$x_i^{k+1} = x_i^k + h_k s_i^k,\ i = 1, \ldots, n,\quad h_k = \arg\min_h f_0(X^k + hS^k),$$

$$S^k = -\nabla f_0(X^k) = \left[-\frac{\partial f_0(X^k)}{\partial x_1}, -\frac{\partial f_0(X^k)}{\partial x_2}, \ldots, -\frac{\partial f_0(X^k)}{\partial x_n} \right].$$

Поиск заканчивается при выполнении следующего условия:

$$\sum_{i=1}^{n}\left[\frac{\partial f_0(X^k)}{\partial x_i}\right]^2 \le \varepsilon, \qquad (2.3)$$

где ε - заранее заданное положительное число.

Для приближенного вычисления частных производных целевой функции $f_0(X)$ по переменным x_i, i=l, ..., n, в точке X^k используется разностная схема:

$$\frac{\partial f_0(X^k)}{\partial x_i} = \frac{f_0(x_1, x_2, ..., x_i + \Delta x, ..., x_n) - f_0(x_1, x_2, ..., x_n)}{\Delta x}$$

где Δx - малое приращение по переменным x_i, i=1,..., n.

Геометрическая иллюстрация работы метода наискорейшего спуска приведена на рис.2.11.

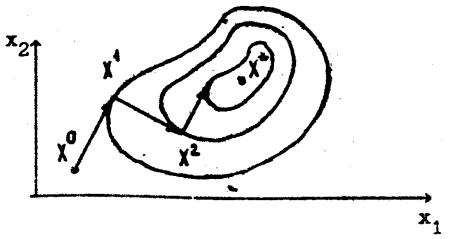


Рис 2.11 **Метод наискорейшего спуска**

Метод сопряженных градиентов

Функция n переменных, приводимая к виду

$$f_0(X) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_i x_j + \sum_{i=1}^{n} b_i x_i + c, \qquad (2.4)$$

называется квадратичной функцией. В векторной форме записи функцию $f_0(X)$ можно представить в виде

$$f_0(X) = \frac{1}{2}XAX^T + BX^T + c,$$

где $X=(x_1, x_2, \ldots, x_n)$ - вектор-строка; X^T - вектор-столбец; T- символ транспонирования; $A=(a_{ij})$, i, j=l,..., n – симметричная матрица размера $n \times n$; $B=(b_i)$, i=1,…, n - постоянный вектор размера n, c - константа.

Для квадратичных функций

$$\frac{\partial f_0}{\partial x_i} = \sum_{j=1}^{n} a_{ij}x_j + b_i, \quad i = 1,..., n,$$

$$\frac{\partial^2 f_0}{\partial x_i \partial x_j} = a_{ij}, \quad i, j = 1,..., n$$

Идея метода сопряженных градиентов основана на стремлении минимизировать квадратичную функцию за конечное число шагов. Для этого требуется найти направления $S^0, S^1, \ldots, S^{n-1}$ такие, что последовательность n одномерных минимизаций вдоль этих направлений приводит к отысканию минимума функции (2.5) при любом начальном приближении X^0.

Указанным свойством обладает система взаимно сопряженных относительно матрицы вторых производных A векторов. Два вектора S^k и S^{k+1} называются сопряженными (относительно матрицы A), если они отличны от нуля и для них выполняется условие
$(S^k)A(S^{k+1})^T=0$.

В методе сопряженных градиентов система взаимно сопряженных направлений строится по правилу

$$S^0 = -\nabla f_0(X^0),$$

$$S^k = -\nabla f_0(X^k) + \beta_{k-1} S^{k-1}, \quad k = 1,2,....,$$

где коэффициент β_{k-1} определяется из условия сопряженности векторов S^k и S^{k-1}, т.е.

$(S^{k-1})A(S^k)^T=0$

$(S^{k-1})A(-\nabla f_0(X^k) + \beta_{k-1} S^{k-1})^T = 0.$

Проведя несложные преобразования, получим выражение для вычисления β_{k-1} при минимизации квадратичных функций:

$$\beta_{k-1} = \frac{(S^{k-1})A(\nabla f_0(X^k))^T}{(S^{k-1})A(S^{k-1})^T} \qquad (2.6)$$

Для произвольной k-й итерации следующее приближение X^{k+1} к точке минимума X^* определяется из условия того, что точка X^{k+1} является точкой минимума функции $f_0(X)$ вдоль направления, определяемого вектором S^k, т.е.

$$X^{k+1}=X^k + h_k S^k, \quad h_k = \arg\min_h f_0(X^k + hS^k).$$

Локализация отрезка (h', h") содержащего значение h_k, и поиск значения h_k выполняются с использованием тех же процедур, что и в методе наискорейшего спуска.

Для вычисления β_{k-1} удобнее пользоваться другим выражением, в котором отсутствует матрица вторых производных A:

$$\beta_{k-1} = \frac{(\nabla f_0(X^k))(\nabla f_0(X^k))^T}{(\nabla f_0(X^{k-1}))(\nabla f_0(X^{k-1}))^T} \qquad (2.7)$$

В таком виде оно может быть использовано как для минимизации квадратичных, так и неквадратичных функций, у которых значения вторых производных в точках X^k, k=0,1,…, не являются неизменными.

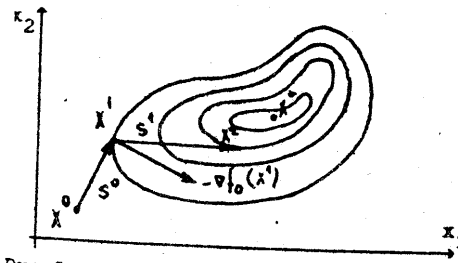


Рис 2 12 Метод сопряженных градиент

2) Классификация центральных процессоров Intel и соответствующих локальных и системных шин ПЭВМ типа IBM PC.

Классификация:

- 4-битные процессоры (4004: первый процессор, реализованный в одной микросхеме, 4040) 1971-1972 г.;
- 8-битные процессоры (8008, 8080, 8085) 1972-1976 г.;
- 16-битные процессоры: Происхождение x86 (8086, 8088, 80186, 80188, 80286) 1978-1982 г.;
- 32-битные процессоры: Линия 80386 (80386DX, 80386SX, 80376, 80386SL, 80386EX) 1985-1990 г.;
- 32-битные процессоры: Линия 80486 (80486DX, 80486SX, 80486GX, 80486DX2, 80486SL, 80486DX4) 1989-1994 г.;
- 32-битные процессоры: Pentium I (Pentium («Классический»), Pentium MMX) 1993, 1997;
- 32-битные процессоры: микроархитектура P6/Pentium M (Pentium Pro, Pentium II, Celeron (Pentium II-based), Pentium III, Pentium II и III Xeon, Celeron (Pentium III базирующийся на ядре Coppermine), Celeron (Pentium III на ядре Tualatin), Pentium M, Celeron M, Intel Core,

Pentium Dual-Core, Dual-Core Xeon LV) 1995-1999 г.;
- 32-битные процессоры: микроархитектура NetBurst (Pentium 4, Xeon) 2000-2001 г.;
- 64-битные процессоры: IA-64(Itanium, Itanium 2, Itanium 2 серия 9000, Itanium 2 серия 9100) 2001 г.;
- 64-битные процессоры: EM64T — Микроархитектура NetBurst (Pentium 4F, D0 и более поздние степпинги, Pentium D, Pentium Extreme Edition, Xeon) 2005 г.;
- 64-битные процессоры: EM64T — Микроархитектура Intel Core (Xeon, Intel Core 2, Pentium Dual Core, Celeron Dual Core, Celeron (микроархитектура Core), Celeron M (микроархитектура Core)) 2007-2010 г.;
- Intel Atom (32-битные процессоры: IA-32, 64-битные процессоры: EM64T) 2008 г.;
- 64-битные процессоры: EM64T — Микроархитектура Nehalem (Intel Celeron, Intel Pentium, Intel Core i7, Intel Core i5, Intel Core i3, Intel Core i7 Extreme Edition, Intel Xeon (UP/DP), Intel Xeon MP) 2011 г.;
- 64-битные процессоры: EM64T — Микроархитектура Sandy Bridge (Intel Celeron, Intel Pentium, Intel Core i3, Intel Core i5, Intel Core i7, Intel Core i7 Extreme Edition, Intel Xeon E3);
- 64-битные процессоры: EM64T — Микроархитектура Ivy Bridge (Intel Core i3, Intel Core i5, Intel Core i7) 2012 г..

В компьютерах PC/AT стала применяться 16-разрядная системная шина ISA (Industry Standart Architecture). С появлением процессоров 80386, а затем и 80486 системная шина ISA стала тормозом на пути увеличения общей производительности системы. Новой шиной стала EISA (Extended Industry Standart Architecture). В пакетном режиме она обеспечивала теоретическую скорость 33 Мб/с. Однако, данная шина не получила большого распространения и использовалась преимущественно в мощных файл-серверах и рабочих станциях. Шина PCI является надстройкой над локальной шиной процессора, т.к. связана с ней логически и электрически с помощью специального контроллера - «моста» (bridge), благодаря чему одновременно к шине может быть подключено до 10 устройств.

3) Реалистическая графика. Обратная трассировка луча. Полная модель освещения Уиттеда.

С развитием вычислительной техники и ее возможностей все больший интерес вызывает задача построения по заданной модели сцены изображения, максимально приближенного к реальному. Построенное таким образом изображение должно восприниматься человеком как настоящее, реалистическое.

Для избавления от излишних вычислений используется обратная трассировка, в которой вычисляются интенсивности только лучей, попавших в глаз наблюдателя.

Идея обратной трассировки лучей:
Для определения цвета пиксела экрана через него из камеры проводится луч, ищется его ближайшее пересечение со сценой и определяется освещенность точки пересечения. Эта освещенность складывается из отраженной и преломленной энергий, непосредственно полученных от источников света, а также отраженной и преломленной энергий, идущих от других объектов сцены. После определения освещенности этой точки учитывается ослабление света при прохождении через прозрачный материал и в результате получается цвет точки экрана.

Для определения освещенности, привносимой непосредственным освещением, из точки пересечения выпускаются лучи ко всем источникам света и определяется вклад всех источников, которые не заслонены другими объектами сцены. Для определения отраженной и преломленной освещенности из точки выпускаются отраженный и преломленный лучи и определяется привносимая ими освещенность.

Простейшей моделью освещенности является модель Уиттеда, согласно которой световая энергия, покидающая точку в направлении вектора, определяется формулой

$$I(\lambda) = k_a I_a(\lambda) + k_d C(\lambda) \sum_j I_j(\lambda)(n, l_j) + k_r I_r(\lambda) e^{-\beta_r d_r} + k_s \sum_j I_j(\lambda)(n, h_j)^p + k_t I_t(\lambda) e^{-\beta_t d_t}$$

Где:

- k_a, k_d,k_s- веса фонового, диффузного и зеркального освещения;
- k_r, k_t - веса отраженного и преломленного лучей;
- C(l) - цвет объекта в точке P;
- n - вектор нормали в точке P;
- l_j - направление на j-е источник света;
- I_j(l) - цвет j-го источника света;
- b_r , b_t - коэффициенты ослабления для сред, содержащих отраженный и преломленный лучи;
- d_r , d_t - расстояния от точки P до ближайших точек пересечения отраженного и преломленного лучей с объектами сцены;
- h_j - вектор, задаваемый формулой
- p - коэффициент Фонга.