Государственный комитет Российской Федерации
по высшему образованию

Самарский государственный
аэрокосмический университет
имени академика С.П. Королева

В.Н. Гаврилов

# ОСНОВЫ САПР

Учебное пособие

Самара 1994

УДК 681.3

Основы САПР: Учебн. пособие /В.Н. Г а в р и л о в.
Самар. аэрокосм. ун-т. Самара , 1994. 59 с.
ISBN 5-230-16860-9

В пособии дается краткое изложение главных разделов курса "Основы САПР". Наиболее полно описаны элементы математического обеспечения САПР, формализован процесс проектирования, рассмотрены вопросы моделирования и методы решения оптимизационных задач.

Пособие предназначено для студентов третьего курса и знакомит читателя с назначением систем автоматизированного проектирования, применяемой терминологией и методами. Выполнено на кафедре конструкции и проектирования летательных аппаратов.

Табл. 3. Ил. 22. Библиогр.: II назв.

Печатается по решению редакционно- издательского совета Самарского государственного аэрокосмического университета имени академика С.П. Королева

Рецензенты: Н.И. С о с н и н, А.В. С о л л о г у б

ISBN 5-230-16860-9

## I. ПРОЕКТИРОВАНИЕ И АВТОМАТИЗАЦИЯ

### I.I. Обоснование САПР

Говоря о высоких темпах развития техники, мы не всегда представляем себе количественные характеристики этого развития. Вот некоторые данные [I] по темпам удвоения количественных показателей, характеризующих внедрение техники в нашу жизнь:

- число различных классов технических систем - I0 лет;
- число деталей в изделиях одного назначения - I5 лет;
- объем научно-технической информации - 8 лет;
- объем проектно-конструкторских работ - 3 года.

Чтобы представить себе значительность этих темпов, вспомним известную историю о мудреце, попросившем за изобретение шахмат "скромную" награду: на первую клетку шахматной доски положить одно зерно, а на каждую следующую - в два раза больше. На последней клетке должно быть $2^{63}$, что примерно равно $10^{19}$ зерен (мировое производство зерна за несколько столетий).

Несложно подсчитать, что за 30 лет объем проектно-конструкторских работ должен возрасти в I000 раз. До сих пор эта проблема решалась увеличением числа специалистов (при этом производительность труда конструкторов росла незначительно - на 20% с 1900 по 1960 г.). Увеличение числа работников требует координации их усилий, что приводит к новому увеличению численности исполнителей (управляющего персонала). Поэтому возможности выполнения возрастающего объема работ путем увеличения числа конструкторов уже в шестидесятых годах были исчерпаны. Появление и развитие ЭВМ позволяет найти выход из создавшегося противоречия.

В конце 60-х годов ЭВМ начинают широко применяться для выполнения сложных расчетов в конструкторских бюро.

Задачи, решаемые в проектных организациях, усложняются с увеличением быстродействия и памяти ЭВМ и развитием математических методов оптимизации. И наконец, в семидесятых годах, с появлением развитых средств ввода и вывода информации при работе с ЭВМ, стал возможен переход к комплексной автоматизации проектных работ. Создание систем автоматизированного проектирования (САПР) требует многих лет и больших усилий. Оно продолжается и далеко до завершения. Однако опыт, накопленный к настоящему времени, позволяет обобщить полученные результаты и наметить дальнейшие пути развития САПР.

Автоматизированное проектирование (проектирование в режиме взаимодействия человека и ЭВМ) – промежуточное звено между традиционным (ручным, безмашинным) и автоматическим проектированием.

## I.2. Требования к САПР

Приступая к созданию САПР, мы интуитивно стремимся механизировать интеллектуальный труд проектировщика. При этом необходимо наиболее полно использовать возможности ЭВМ (высокую скорость обработки информации, возможность длительного хранения больших объемов информации, программное управление). Эти возможности при рациональном их использовании дают выигрыш в следующих характеристиках проектирования:

- качество проектирования повышается вследствие увеличения числа просматриваемых вариантов и выбора оптимального, применения более адекватных и точных моделей, использования наглядного инструмента для анализа и контроля;

- сроки проектирования сокращаются вследствие автоматизации информационных потоков, механизации непроизводительных операций и ускорения расчетов;

- стоимость проекта снижается вследствие замены натурных экспериментов на численные с соответствующим сокращением объема испытаний и доводок, сокращения информационных и чертежных подразделений;

- методы проектирования получают дальнейшее развитие вследствие большей информированности исполнителей, упорядочения документации, формализации проектных процедур.

Для достижения планируемых целей САПР должны удовлетворять требованиям проектных организаций и общесистемным требованиям.

Требования проектных организаций включают:

- адекватность[1] (соответствие) САПР объекту и процессу проектирования;
- надежность системы (как вероятность получения безошибочного результата к заданному сроку);
- эффективность (экономия трудозатрат);
- удобство работы с системой.

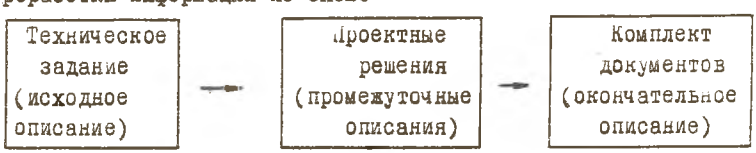
К системным требованиям относятся:

- интеграция системы (суммирование всех проектных операций или процедур, образующих завершенный цикл проектирования);
- информационная обеспеченность (возможность быстрого и удобного получения необходимой информации);
- модульность (разбиение системы на функциональные фрагменты);
- эволюционность (приспособленность к изменениям и доработкам).

## I.3. Процесс проектирования

Построение САПР невозможно без формализации (описания на математическом и логическом языках) процесса проектирования. Полная формализация привела бы к созданию систем автоматического проектирования, но современное состояние науки не позволяет описать творческие операции, без которых проектирование невозможно. (Вопрос о принципиальной возможности формализации творчества остается открытым).

С точки зрения САПР проектирование представляет собой процесс переработки информации по схеме




Проектные решения – результаты выполнения проектных п р о ц е д у р. Составными частями проектных процедур являются проектные о п е р а ц и и.

---

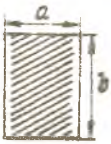
[1] Соответствующий, верный, точный (от лат. *adaequatus* – правильный

Проектные процедуры и операции направлены на решение задач анализа и синтеза. А н а л и з[1] - исследование объектов и явлений путем рассмотрения отдельных сторон, свойств, частей. Анализ в САПР - проектная процедура получения информации о свойствах проектируемого объекта. С и н т е з[2] - метод исследования объектов (явлений) в единстве и взаимной связи его частей, обобщение, сведение данных в единое целое. Синтез в САПР - проектная процедура получения нового описания объекта. Синтез находится в диалектическом единстве с анализом: при синтезе создаются, а при анализе оцениваются описания проектируемых объектов.

Пример 1. Спроектировать балку постоянного сечения.


$M_{max} = P\ell;\ \sigma_{доп} \geq M_{max}/W.$

Задача анализа: задаемся сечением прямоугольной формы $a \times b$ и проверяем условие прочности




$W = ab^2/6;$

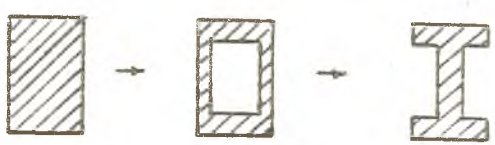
$\sigma_{доп} \geq 6P\ell/(ab^2).$

Задача параметрического синтеза. Подобрать размеры $a$ и $b$ так, чтобы выполнялись условия прочности и дополнительные ограничения $a \leq a_{max}$, $b \leq b_{max}$:

$b = b_{max}; \quad a = 6P\ell/\sigma_{доп} b_{max}^2 \leq a_{max}.$

Задача структурного синтеза. Выбрать форму и размеры сечения из условия минимизации массы балки:

$m = \rho S\ell; \quad \rho = const; \quad \ell = const; \quad S \to min.$

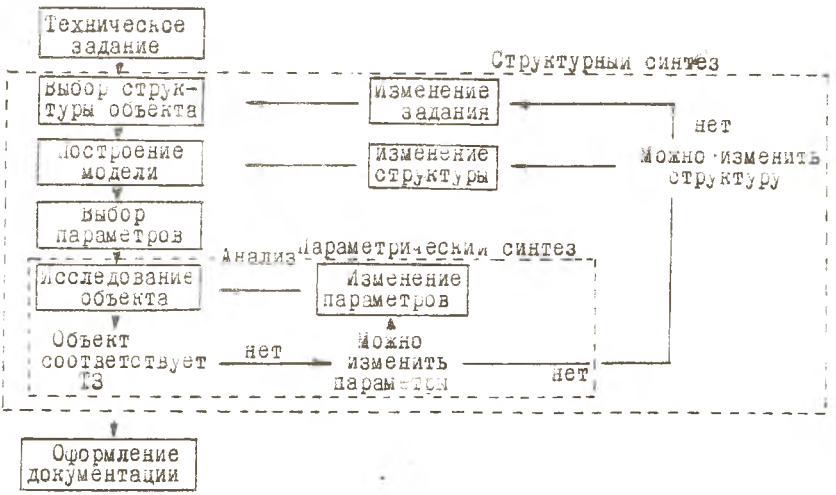


---

[1] от греч. *analysis* - разложение.

[2] от греч *synthesis* - соединение.

Как показывает приведенный пример, процесс проектирования строится по следующей схеме (рис. I).




Р и с. I. Схема процесса проектирования

Как видно из приведенной схемы, параметрический синтез многократно включает в себя задачу анализа. Структурный синтез требует многократного повторения параметрического синтеза. Проектирование каждого элемента требует решения задач анализа, структурного и параметрического синтеза. Если трудоемкость решения задач анализа возрастает пропорционально росту числа элементов, то трудоемкость решения задач синтеза представляет собой степенную функцию от числа элементов (рис. 2).


Р и с. 2. Зависимость трудоемкости решения задач от числа элементов

Именно эти зависимости и порождают кризисные явления в проектировании, требующие разрешения противоречия "срок разработки - качество" и вынуждающие повышать производительность труда конструкторов путем применения САПР.

I.4. Основы системного подхода

Под с и с т е м о й понимают множество взаимосвязанных объектов (элементов). Для определения системы важно выделить п р и з н а к и составляющих ее объектов и связи между ними. О к р у ж е н и е - множество объектов вне системы, влияющих на систему или испытывающих ее влияние. Задача выделения системы и окружения (в общем случае) не является тривиальной. В окружение, в частности, должны быть включены: окружающая среда, технологические, экономические, организационные факторы.

Важными свойствами системного подхода являются д е к о м п о з и ц и я и и е р а р х и ч н о с т ь. В каждом элементе можно выделить части (его составляющие). В свою очередь, каждая система является элементом (подсистемой) объединения более высокого ранга (иерархического уровня).

Пример 2.


В том случае, когда осуществляется проектирование системы, в поле зрения проектировщика попадает только три уровня:

- проектируемая система;
- система верхнего уровня (как элемент окружения);

- подсистемы (как неделимые элементы системы).

Важное свойство системы - ее развитие во времени. Фактор времени должен учитываться при проектировании в нескольких аспектах:

- предыстория системы (прототипы, тенденции развития);
- современный уровень развития техники (выбор соответствующих методов и технических решений);
- определение рациональной долговечности системы (с учетом морального старения и стоимости).

С методологической точки зрения системный подход - это способ организации мышления при проектировании. Его применение в полной мере оправдано только при поиске о п т и м а л ь н ы х проектных решений.

## I.5. Принципы проектирования

Многолетний опыт проектирования технических систем позволил сформулировать и успешно применять ряд принципов, определяющих методы проектирования.

Д е к о м п о з и ц и я и и е р а р х и ч н о с т ь - эти принципы вытекают из системного подхода. Описания объекта разбиваются в соответствии с возможностями восприятия, что позволяет упростить проектные процедуры и расширить фронт работы. Разбиение проводится как по степени подробности (иерархические уровни объекта), так и по характеру свойств объекта (функциональный, конструкторский, технологический и другие аспекты).

Т и п и з а ц и я и у н и ф и к а ц и я - принципы, позволяющие совершенствовать и многократно применять готовые технические решения и методы.

М н о г о э т а п н о с т ь и и т е р а ц и о н н о с т ь проектирования являются следствиями перечисленных выше принципов и дают возможность многократно применять типовые проектные процедуры, последовательно уточняя решение.

Перечисленные принципы успешно применяются в системах автоматизированного проектирования.

## I.6. Структура САПР

Система автоматизированного проектирования включает:

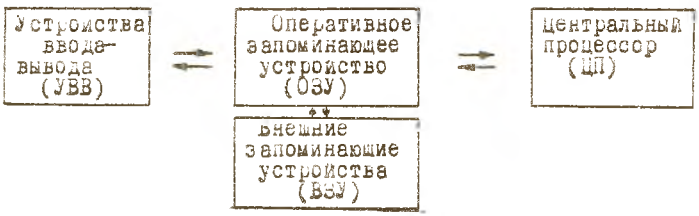
- техническое обеспечение;
- математическое обеспечение;

- программное обеспечение;
- информационное обеспечение;
- лингвистическое обеспечение;
- методическое обеспечение;
- организационное обеспечение.

## 2. Технические средства САПР

### 2.1. Структура ЭВМ

За пятьдесят лет, прошедшие с момента создания ЭВМ, основная структура вычислительных машин осталась неизменной (рис. 3).




Р и с. 3. Основная структура вычислительных машин

Для выполнения вычислительного процесса необходимо иметь исходные данные и предписание, определяющее действия над данными. Перечисленная информация должна определенное время сохраняться в машине. Проведение вычислений подразумевает наличие устройства, способного выполнять ряд операций над данными. Наконец, необходимы устройства, способные принять информацию и сообщить результат человеку. В соответствии с перечисленными функциями в состав ЭВМ входят: память, процессор, внешние устройства (см. рис. 3).

Память ЭВМ – устройства, предназначенные для записи, хранения и извлечения информации[1]. Память ЭВМ включает оперативное запоминающее устройство (ОЗУ) и внешние запоминающие устройства (ВЗУ). ОЗУ используется для хранения непосредственно обрабатываемой информации, ВЗУ – для длительного хранения большого количества информации.

---

[1] Информация (лат. – разъяснение, изложение) – некоторые сведения, совокупность данных, знаний.

Процессор предназначен для выполнения операций над данными. во время работы ЭВМ процессор постоянно обменивается информацией с ОЗУ.

Загрузка программ и данных осуществляется с помощью устройств ввода, выдача результатов - с помощью устройств вывода.

Перечисленные устройства с развитием ЭВМ значительно усложнились. Так появилось несколько процессоров, управляющих не только процессом вычисления, но и передачей данных; выделились отдельные функциональные блоки оперативной памяти и особенно усложнились каналы передачи информации, став самостоятельными сложными электронными устройствами (на схеме они показаны просто стрелками). Приведенная упрощенная схема дает наглядное представление о принципах устройства и работы ЭВМ, и ее следует рассматривать как первое приближение.

## 2.2. Память ЭВМ

### 2.2.I. Количество информации

Сообщение может быть записано набором символов (букв, цифр, знаков). Если алфавит, используемый для передачи сообщения, содержит $m$ символов, то сообщение из одного символа может быть записано $m$ способами, сообщение из двух символов - $m^2$ способами, из $n$ символов - $m^n$ способами. Таким образом, слово из $n$ символов может передать $m^n$ различных сообщений. Очевидно, что с ростом числа возможных сообщений количество информации в одном сообщении должно расти. Информация, содержащаяся в единственно возможном сообщении, равна нулю. С точки зрения удобства использования, количество информации должно быть пропорционально числу символов в сообщении. С учетом перечисленных требований зависимость, выражающая количество информации от числа возможных сообщений, выбрана в виде

$$S = \log m^n = n \log m.$$

Наименьшее не равное нулю количество информации содержится в сообщении, определяющем один из двух возможных вариантов. Это количество информации принимается за единицу измерения

$$S_0 = \log_2 2 = 1 \text{ бит.}$$

(Примеры: ответ на вопрос, допускающий лишь два ответа - "да"

или "нет"; цифра числа, записанного в двоичной системе счисления).

### 2.2.2. Устройство памяти

Физический элемент памяти должен сохранять минимальное количество информации (I бит). Такой элемент легко реализуется многими электронными устройствами. Наиболее распространенной до последнего времени была память на магнитных сердечниках. Возможны и другие виды реализации оперативных запоминающих устройств (полупроводниковые, на магнитных пленках, лазерные, голографические и др.).

Элементы памяти объединены в блоки, называемые я ч е й к а м и. Размер ячеек на разных ЭВМ различен (наиболее часто в ячейке содержится 32 бита).

### 2.2.3. Кодирование информации

Наиболее привычная форма обмена информацией – письменные сообщения, использующие буквенный алфавит и цифры десятичной системы счисления. Для записи этой информации в память ЭВМ необходимо установить соответствие между общепринятыми символами и числами (с л о в а м и) в двоичной системе счисления. Представление символов одного алфавита символами другого называется к о д и р о в а н и е м, а таблицы соответствий – к о д о м.

Пусть алфавит содержит 32 буквы, тогда одна буква содержит количество информации $\log 2^5 = 5$ бит. Следовательно, для записи одной буквы этого алфавита в двоичном коде понадобится пять двоичных разрядов. В качестве единицы записи информации в памяти ЭВМ принято восьмиразрядное слово – I байт = 8 бит. Слово, содержащее один байт информации, может быть записано $2^8 = 256$ способами. Такой размер записи позволяет с некоторым запасом закодировать все общепринятые символы (буквы русского и латинского алфавитов, цифры, математические знаки).

Пример одного из используемых в ЭВМ кодов:

| | |
|---|---|
| А – II000001, | I – IIII0001, |
| Б – II000010, | 2 – IIII0010, |
| ... | ... |

### 2.2.4. Запись данных в памяти ЭВМ

Все данные записываются в ЭВМ в закодированном виде. При этом символы преобразуются с помощью специальных кодов. Числа пере-

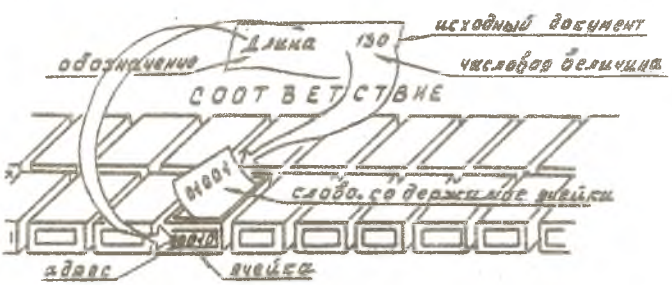
водятся в двоичную систему счисления. При записи числа в память ЭВМ возможны две формы его представления.

| Наименование | Форма представления | Запись в десятичной системе |
|---|---|---|
| – с фиксированной запятой (целое) | знак числа \| число | –27<br>+ 285 |
| – с плавающей запятой (вещественное) | знак порядка \| порядок \| знак числа \| мантисса | 35,28I<br>2I58·I0$^{-3}$ |

Запись числа с плавающей запятой более универсальна, но требует большего объема памяти (увеличивается количество информации) и большего времени обработки. Совместное применение двух форм записи чисел позволяет экономить объем памяти ЭВМ и увеличивать быстродействие.

Физическим элементом записи в память ЭВМ является я ч е й к а. Все ячейки последовательно пронумерованы. Номер ячейки называется а д р е с о м. Адреса используются для распознавания данных при считывании и записи.

Для иллюстрации организации памяти можно привести такую аналогию (рис. 4). Склад (ОЗУ) заполнен пронумерованными ящиками (ячейки с адресами), в ящиках находятся закодированные значения числовых величин, определенных исходным документом. Обозначение числовой величины соответствует адресу ячейки.




Р и с. 4. Графическая аналогия организации памяти

## 2.3. Центральный процессор ЭВМ

Центральный процессор - основная часть ЭВМ, проводящая обработку информации в соответствии с заданной программой и управляющая вычислительным процессом. В состав процессора входят: арифметическо-логическое устройство, предназначенное для выполнения заданного набора операций над данными; устройство управления, предназначенное для управления работой машины; регистровая (локальная) память для временного хранения информации во время ее обработки; система приоритетных прерываний; управляющая память; блок контроля и диагностики; блок связи с ОЗУ; блок защиты памяти и др.

Основные характеристики процессора: состав системы команд, время их выполнения, доступное адресное пространство, способы адресации, разрядность обрабатываемых слов, объем локальной памяти и др.

### 2.3.1. Команда и ее запись

Действия ЭВМ задаются специальными управляющими сигналами - командами. В памяти ЭВМ команда представлена следующим образом:

| код операции | адреса |
|---|---|

Код операции представляет собой условное число, обозначающее выполняемую операцию (например: 01 - сложение, 02 - вычитание и т.д.). Адреса во второй (информационной) части команды указывают место в памяти ЭВМ, где хранятся обрабатываемые данные и результат. В зависимости от числа адресов, указываемых в команде, различают одноадресные, двухадресные и трехадресные ЭВМ.

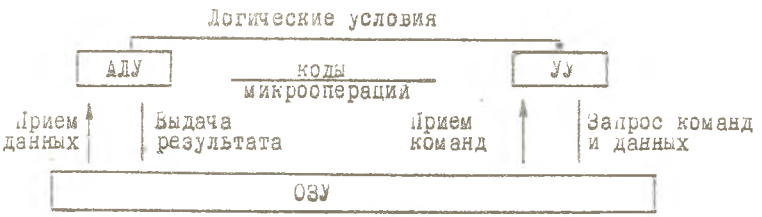
Команды, закодированные в двоичной системе счисления, хранятся в памяти таким же образом, как и числа.

### 2.3.2. Работа процессора

Команды выдаются из памяти машины в устройство управления (УУ) поочередно (или в порядке, указанном в программе). Устройство управления формирует запросы по адресам и микрокоманды, управляющие работой арифметическо-логического устройства (АЛУ). АЛУ получает данные (согласно адресам, приведенным в команде) и

выполняет над ними действия, определенные кодом операции (рис. 5).

Выполнение операции заключается в многократном исполнении соответствующей последовательности микроопераций над разрядами двоичного числа (например: суммирование, сдвиг, пересылка, очистка). Результат операции посылается в оперативную память, в ячейку, адрес которой приведен в соответствующей команде. Из АЛУ в УУ поступают логические условия (например: "число А - отрицательно"), на основании которых УУ изменяет порядок выполнения команд или характер работы машины.




Р и с. 5. Схема работы процессора

## 2.4. Принципы вычислений на ЭВМ

### 2.4.1. Программное управление

Управление вычислениями осуществляется с помощью программы, записанной в оперативную память ЭВМ. Программа представляет собой набор команд.

Применение программного управления позволяет использовать опыт многих людей, а не только человека, непосредственно ведущего вычисления. Контроль за быстро протекающими процессами вычисления недоступен человеку из-за ограниченных возможностей его чувств и может быть осуществлен только с помощью программ. Предварительная загрузка программы в память ЭВМ экономит время процессора во время счета, поскольку считывание команд из ОЗУ происходит намного быстрее, чем их ввод в машину.

### 2.4.2. Дискретное представление и преобразование информации

Любая информация записывается в память ЭВМ в двоичном коде с дискретностью один двоичный разряд. Число разрядов для записи

элементарной порции данных (символа, числа) кратно размеру ячейки.

Обработка информации проводится поразрядно дискретными порциями. До окончания обработки содержимого ячейки (слова) результат операции не определен.

Альтернативой дискретным представлению и обработке информации является непрерывный принцип моделирования, применяемый, например, в аналоговых вычислительных машинах (АВМ). Физические величины в АВМ моделируются величиной тока или напряжения и обрабатываются непрерывно.

### 2.4.3. Адресный принцип идентификации данных

Принцип заключается в том, что каждому элементу данных ставится в соответствие номер ячейки памяти (адрес), в которой хранятся эти данные.

Альтернативой адресному принципу может быть паспортный принцип идентификации. Адресный принцип не требует дополнительных объемов памяти для размещения наименования данных (паспорта).

### 2.4.4. Цикличность обработки информации

Каждое действие над данными выполняется как последовательность повторяющихся циклов микроопераций над разрядами. Любая операция тоже представляет собой цикл:

- выбрать элемент данных из ячейки;
- провести действие;
- послать результат в ячейку памяти.

Организация процесса вычислений в виде циклов позволяет упростить схему ЭВМ, сократить ее элементарную базу, уменьшить вес и размеры.

### 2.4.5. Формализм записи и обработки данных

Информация, какой бы вид она ни имела до кодирования (числа, текст, команды), записывается в память ЭВМ в двоичной системе счисления. По внешнему виду записи невозможно отличить число от команды и для расшифровки требуется дополнительная информация (указание о виде записи, начале и конце). Обработка данных проводится независимо от их содержания. Так команда (по желанию программиста или по ошибке) может быть использована и обработана как число.

Формальный подход к записи и обработке данных позволяет упростить конструкцию ЭВМ.

2.4.6. Использование обратных связей

Применение этого принципа позволяет исключить бессмысленные вычисления (например, деление на ноль), контролировать состояние машины и результаты вычислений на случай появления сбоев или ошибок данных. Примером обратной связи в процессоре является выработка логических условий арифметическим устройством и передача их в устройство управления.

2.5. Характеристики ЭВМ

Быстродействие измеряется количеством операций, выполняемых ЭВМ в секунду. Различные операции над данными ЭВМ выполняет за разное время, поэтому при оценке быстродействия подсчитывается среднее количество операций, выполняемых за секунду. Для современных ЭВМ быстродействие составляет $10^5$...$10^7$ оп/сек.

Объем памяти измеряется в килобайтах - К (К=1024 байт) и мегабайтах - М (М=1024 К). Для современных ЭВМ объем ОЗУ зависит от класса машины и составляет 0,5 ...10 М.

Удобство обмена информацией - качественная характеристика - определяется набором внешних устройств, зависящим от назначения ЭВМ.

3. МАТЕМАТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ САПР

3.1. Этапы решения задачи на ЭВМ

В процессе решения проектной задачи с применением ЭВМ можно выделить следующие этапы:

- постановка задачи;
- формализация задачи; ф о р м а л и з а ц и я - представление и изучение области знания в виде формальной системы (например, в математической форме);
- моделирование; м о д е л и р о в а н и е - исследование объектов (явлений) путем построения и изучения их

моделей[1];

- выбор метода решения;
- алгоритмизация;
- разработка программы;
- получение результата (расчет на ЭВМ);
- анализ результата и принятие решения.

Первые четыре этапа из перечисленных имеют прямое отношение к математическому обеспечению САПР и будут подробно обсуждаться далее.

### 3.2. Постановка задачи

Словесная (вербальная[2]) постановка задачи включает уяснение трех важных вопросов:

- формулировку цели;
- определение средств достижения цели;
- выявление ограничений (или условий), при которых решается задача.

#### 3.2.1. Формальная постановка задачи

Математически задача проектирования может быть сформулирована как оптимизационная задача вида: минимизировать функцию $\Phi(X)$ по переменной $X$ при ограничениях $f_i(X) \leq 0$. Постановка задачи в таком виде включает три момента: выбор функции цели $\Phi(X)$, выбор переменной $X$ и определение ограничений.

#### 3.2.2. Критерий и функция цели

Ф у н к ц и я  ц е л и представляет собой математическую запись критерия проектирования. К р и т е р и й – числовой показатель, характеризующий качество объекта проектирования.

Функция цели аналитически или алгоритмически выражает зависимость критерия качества проектируемого объекта от параметров объекта. При проектировании ЛА наиболее распространенными критериями являются его стоимость (затраты на разработку, производство, эксплуатацию), эффективность (выполняемая работа, прибыль, вероятность вы-

---

[1] Модель – от лат. *modulus* – мера, образец.
[2] От лат. *verbalis* – словесный.

полнения поставленной задачи и др.), надежность (вероятность безотказной работы, безопасность полетов и др.), масса конструкции.

В состав переменных включают параметры объекта, изменяемые при проектировании. Переменные выбираются таким образом, чтобы при минимальном числе параметров были охвачены все существенные стороны проектирования объекта.

Критерии качества не полностью характеризуют свойства объекта. Часть свойств регламентируется в форме условий (ограничений).

### 3.2.3. Многокритериальность

Большое число и разнообразие требований к ЛА приводит к неопределенности цели проектирования, что автоматически делает задачу многокритериальной. Это приводит к трудностям в математической постановке и решении задачи. Для преодоления проблемы многокритериальности иногда пытаются свести разнородные требования к одному критерию, представленному в виде частного (например, эффективность деленная на стоимость) или в виде суммы взвешенных составляющих (например, оценки приращения массы при выполнении какого-либо требования к объекту). Выбор комплексного критерия связан с субъективным назначением весов слагаемых, а получаемые решения не всегда приемлемы.

Часто используют принцип Парето, приводящий к получению "неулучшаемого" решения. Метод, основанный на принципе Парето, для сравнения результатов использует гиперповерхности "равного качества", построенные в пространстве выбранных критериев.

Из других способов разрешения многокритериальности заслуживает внимания упорядочение критериев с заданием уровней притязания [5]. Этот метод отличается своей простотой. Метод сводится к ранжированию критериев и установлению предельных значений каждого из них. Допустимым считается решение, удовлетворяющее системе неравенств

$$\Phi_1(X) \le B_1;$$

$$\Phi_k(X) \le B_k,$$

где $\Phi_1(X)$, ..., $\Phi_k(X)$ – критерии, определяющие качество объекта;

$B_1$, ... $B_k$ – соответствующие предельные значения критериев, полученные методом экспертных оценок.

Получение решения, удовлетворяющего системе неравенств,

сводится к последовательному решению $k$ стандартных оптимизационных задач. На каждой из итераций один из критериев принимается за функцию цели и минимизируется, остальные $(k-1)$ неравенств выступают в роли ограничений.

### 3.3. Математические модели

#### 3.3.1. Общие сведения

Под моделью понимается такая система, которая, отображая объект, способна замещать его и давать новую информацию.

Основные свойства модели: адекватность, точность, удобство использования. Адекватность (соответствие) характеризует модель с качественной стороны (сходство свойств, изоморфизм, аналогия). Точность - количественная характеристика сходства свойств модели и отображаемого физического объекта (или процесса).

Для математических моделей отношение к прототипу устанавливается через м о д е л и р у ю щ у ю  с и с т е м у (математические построения), а важным для рассмотрения свойством являются количественные характеристики.

При разработке математических моделей в САПР важно различать описания п р о ц е с с о в и описания о б ъ е к т о в. Описания процессов включают описания объектов, а также требуют использования физических (или других) закономерностей (еще одной или нескольких промежуточных моделирующих систем).

Например: переход кинетической энергии в потенциальную описывается уравнением $(mV^2)/2 + mgh = const$. Для понимания приведенного уравнения кроме обозначения математических операций (умножения, сложения и т.д.) необходимо определить понятия: масса, скорость, энергия...

Один и тот же объект можно описать различными способами. Например, возможные способы описания круга:

1. Изображение круга в декартовой системе координат.
2. Круг $R=4$, $x_0=0$, $y_0=0$.
3. $x^2+y^2-4<0$.
4. OIIOIIIIIIIIOIIO - рецепторная матрица, описывающая круг с дискретностью I.

### 3.3.2. Примеры построения моделей геометрического тела

Описания геометрических тел часто применяются в САПР, так как без них невозможна формализация объекта проектирования.

Различают две формы представления моделей - входную и внутреннюю. Входная предназначена для ввода описания в ЭВМ, должна быть удобной для пользователя, и поэтому содержит сведения инженерного характера (наименования, размеры, отклонения, координаты). Внутренняя модель предназначена для обработки данных ЭВМ. Она может быть получена из входной машинным расчетом и содержит коэффициенты уравнений, параметры поверхностей, линий и другие числовые данные.

### 3.3.3. Процесс построения модели

Разработка модели геометрического тела методологически разделяется на ряд последовательных этапов;

- расчленение объекта и выделение базовых элементов в соответствии с характером задачи;
- анализ иерархии элементов;
- построение математических моделей базовых элементов;
- анализ структуры объекта и выявление связей;
- объединение математических моделей базовых элементов для построения модели объекта;
- дополнение модели объекта системными параметрами;
- группировка параметров с целью минимизации объема сведений;
- подготовка модели для записи в память ЭВМ (структуризация).

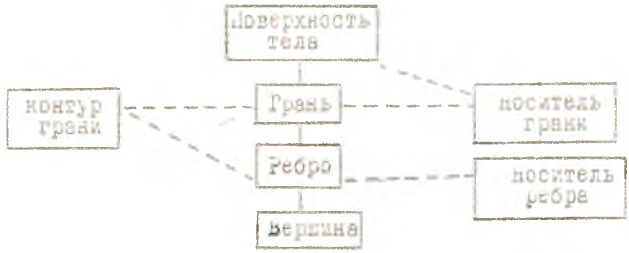
### 3.3.4. Кусочно-аналитическая модель тела

Примем следующие допущения и обозначения:

- тело - замкнутое ограниченное точечное множество;
- поверхность тела - множество граничных точек;
- поверхность тела состоит из множества граней $\{G_i\}$;
- каждая грань принадлежит поверхности $(G_i \in Q_j)$, называемой носителем грани;
- грань ограничена контуром $N_j^i$, состоящим из множества ребер $\{R_j\}$;
- ребро принадлежит линии $(R_j \in B_j)$, называемой носителем ребра;

- вершины $V_k$ - граничные точки ребра.

Иерархия элементов, составляющих тело, представлена графом (рис. 6.).




Р и с. 6. Структура кусочно- аналитической модели

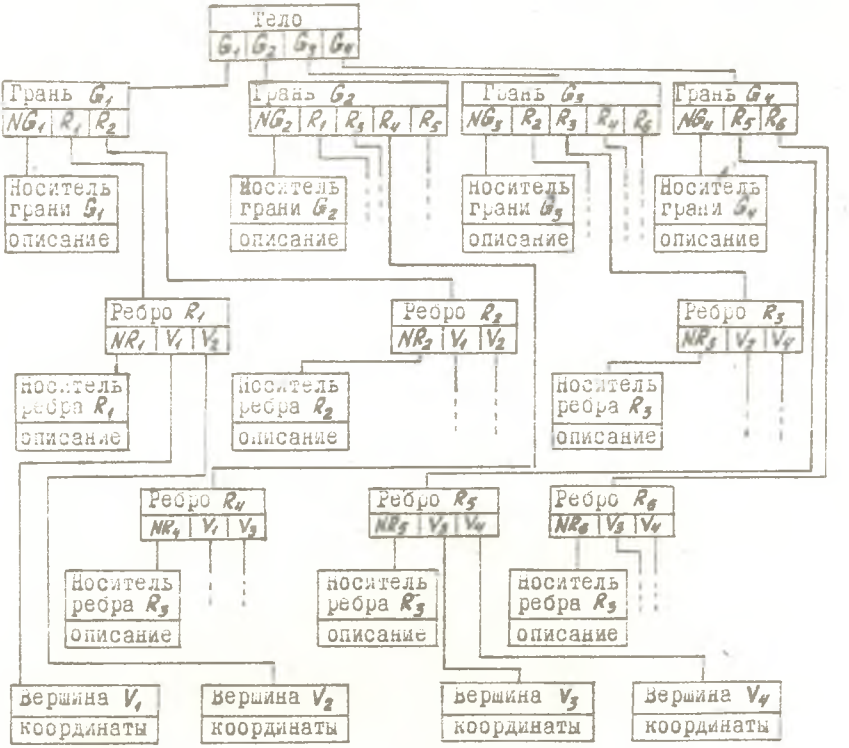
Построим входные модели базовых элементов (грань, ребро, вершина). Вершины (и другие характерные точки) задаются в декартовой системе координат $(x_k, y_k, z_k)$, ребро - носителем ребра и граничными точками. Носитель ребра: тип линии, параметры (например, окружность, $R=50_{мм}$, центр в мм (20,0,0)). Граничные точки задаются номерами. Описание грани включает описание носителя грани (тип поверхности, параметры) и номера ребер, составляющих граничный контур. Пример описания поверхности (носителя грани): цилиндр, $R=50_{мм}$, центр нижнего основания в мм (-20,0,0), центр верхнего основания в мм (60,0,0). Пример структуры данных приведен на рис. 7.

Пример построения массивов данных, описывающих тело, представлен рис. 8..

- $VK$ - массив координат характерных точек;

| VK | | | |
|---|---|---|---|
| $N$ | $X$ | $Y$ | $Z$ |
| I | 50 | 0 | 20 |
| . | ... | ... | ... |

- $NG$ - массив описаний поверхностей - носителей граней;




Р и с. 7. Структура описания тела


Р и с. 8. Изображение детали

| NG | | |
|---|---|---|
| № | Наименование | Параметры |
| I | плоскость | I,2,5 |
| . | ... | ..... |

- NG – массив описаний линий – носителей ребер;

| NR | | |
|---|---|---|
| № | Наименование | Параметры |
| I | окружность | 50,3 |
| . | .... | ..... |

- R – массив описания ребер (номер носителя, номера граничных вершин);
- G – массив описания граней (системные параметры, номер носителя, номера ребер);
- T – массив описания тела (системные параметры, номера граней).

(Системные параметры – специальная информация геометрического или негеометрического характера, например, параметры собственной системы координат грани или чистота обработки грани. Структура внутренней (машинной модели) тела повторяет структуру входной модели. Описание внутренней модели получается программным путем в соответствии с последующей обработкой.

### 3.3.5. Алгебрологическая модель геометрического тела

Кусочно-аналитическая модель трехмерного объекта базируется на элементах "грань, ребро, вершина" и более громоздка в записи по сравнению с логико-аналитической моделью. Рассмотрим логико-аналитическую модель отсека, в которой базовым элементом является поверхность.

Всякая поверхность, заданная уравнением $f(x, y, z) \equiv 0$, делит пространство на две области, определяемые неравенствами

$f(x, y, z) < 0$ и $f(x, y, z) \geq 0$.

Для проверки принадлежности точки $(x_0, y_0, z_0)$ области пространства достаточно подставить координаты точки в уравнение поверхности. Условимся считать область пространства, принадлежа-

щую описываемому телу, положительной. Это условие заставляет нас выбирать знаки коэффициентов уравнения оболочки тела таким образом, чтобы при подстановке координат точки, лежащей "внутри" прибора или отсека, функция $f(x,y,z)$ принимала положительные значения.

Тела представляют собой точечные множества, поэтому для описания связей могут быть использованы элементы теории множеств. Модель отсека использует три операции над множествами:

$\bar{A}$ – абсолютное дополнение множества A;

$A \cup B$ – объединение множеств A и B;

$A \cap B$ – пересечение множеств A и B.

Точечные множества образованы элементами $U = (x, y, z)$, тогда

$$\bar{A} = \{U \mid U \in A\},$$

$$A \cup B = \{U \mid U \in A \text{ или } U \in B\};$$

$$A \cap B = \{U \mid U \in A \text{ и } U \in B\}$$ (см. рис. 9).

Пусть множество A ограничено поверхностью $f_\alpha(x, y, z) = 0$, множество B – поверхностью $f_\beta(x, y, z) = 0$. Определим предикаты[1] (двузначные функции непрерывных переменных) $T_\alpha(x, y, z)$, $T_\beta(x, y, z)$, характеризующие принадлежность точки $U$ множествам A и B соответственно:

$$T_A(U) = \begin{cases} 0, & \text{если } f_\alpha(U) < 0; \\ 1, & \text{если } f_\alpha(U) \geq 0; \end{cases}$$

$$T_B(U) = \begin{cases} 0, & \text{если } f_\beta(U) < 0; \\ 1, & \text{если } f_\beta(U) \geq 0. \end{cases}$$

Введение предикатов позволяет описать отсек с помощью логических (булевых) функций. Для их построения поставим в соответствие операциям над множествами логические операции над предикатами:

---

[1] В математической логике – пропозициональная функция, т. е. выражение, которое при подстановке значений переменных принимает "истинное" или "ложное" значения; другие значения слова п р е д и к а т (от лат. *praedicatum* -сказанное): в узком смысле – свойство, в более широком – отношение.

$T_{\bar{A}}(U) \equiv \neg T_A(U)$ – отрицание (не),

$T_{A\,B}(U) \equiv T_A(U) \vee T_B(U)$ – дизъюнкция (или),

$T_{A\,B}(U) \equiv T_A(U) \wedge T_B(U)$ – конъюнкция (и).

Значения приведенных логических операций определены следующей таблицей:

| $T_A$ | $T_B$ | $\neg T_A$ | $T_A \vee T_B$ | $T_A \wedge T_B$ |
|---|---|---|---|---|
| 0 | 0 | I | 0 | 0 |
| 0 | I | I | I | 0 |
| I | 0 | 0 | I | 0 |
| I | I | 0 | I | I |

Возможности применения логических операций для описания геометрических тел приведены на рис. 9.


Р и с. 9. Операции над множествами

Логические функции принимают лишь два значения 0 и I. Этого достаточно для того, чтобы охарактеризовать принадлежность любой точки пространства одной из двух областей – свободной для размещения (0) или запрещенной (I). Согласно правилу знаков, принятому ранее, "внутри" любого из описываемых тел логическая функция принимает значение I, что соответствует описанию размещаемых тел (область занята), но не соответствует описанию области размещения. С учетом приведенных рассуждений, логическая функция, моделирующая тело, записывается в следующем виде:

$$T_0 = \varphi(T_i) \quad ,$$

где $T_0$ – предикат, определяющий тело;

$\varphi$ – логическая функция;

$T_i$ – предикат, соответствующий поверхности $\ell_i$ .

Пример. Описать с помощью логических функций тело, изображенное на рис. 8.

Тело состоит из одного фрагмента, ограниченного цилиндрической

поверхностью $f_1$ и плоскостями $f_2$, $f_3$, $f_4$. Фрагмент, образующий тело, представляет собой пересечение перечисленных поверхностей. Логическую функцию запишем следующим образом:

$$T_0 = T_1 \wedge T_2 \wedge T_3 \wedge T_4 .$$

Данные, описывающие тело, задаются в виде списка параметров, содержащего слова и числа. Структура списка повторяет структуру тела. Массив данных иерархически упорядочен и состоит на каждом уровне из двух частей: системной и параметрической. Системная информация содержит заголовок, данные о логических связях и числовые данные, относящиеся ко всем элементам уровня.

Модель тела представлена в следующей форме:

$$MT = \{SPt,\ \{MB\}\,S\},\quad S = 1, 2, \ldots ;$$
$$MB = \{SPB,\ PA\};$$

где МТ – модель тела;

МВ – модель базового элемента (поверхности);

*SPt, SPB* – системные параметры тела, базового элемента соответственно;

РА – список параметров, позволяющих вычислить коэффициенты уравнения поверхности;

*S* – номер базового элемента, входящего в состав тела.

Работа с моделью становится более удобной, если при пользовании ею ввести простые правила: вместо записи операции отрицания задаются координаты произвольной точки и обозначается ее положение относительно фрагмента (ВНУТРИ, СНАРУЖИ); запись СНАРУЖИ для точки, расположенной внутри фрагмента, соответствует операции отрицания. За основной базовый элемент принята поверхность. Задание поверхностей коэффициентами уравнений неудобно для конструктора, так как требует проведения предварительных вычислений. При описании тела конструктор идет "от чертежа", поэтому все используемые параметры, определяющие поверхность, должны быть угловыми или линейными размерами. Направление осей, определяющих положение поверхностей в пространстве (например, осей вращения), может задаваться проекциями углов на координатные плоскости или координатами векторов. Первый способ задания требует меньшего числа параметров (примерно на 10%), но информация разнородна по характеру (координаты начальной точки, проекции угла, номера плоскостей, в которых заданы проекции). Угловые размеры не всегда могут быть

непосредственно сняты с чертежа, что позволяет считать координатный способ задания направления осей более удобным. Тип поверхности задаем словом в привычной для конструктора форме. Описание входной информации значительно упрощается, если к базовым элементам отнести наряду с поверхностями простые тела (примитивы), образованные несколькими поверхностями. Входная модель примитива аналогична модели поверхности. Логическая связь поверхностей в примитиве определяется его типом и не требует специального задания. Для описания более сложных форм оболочки отсека возможно задание поверхности в том виде, который принят в конструкторском бюро. Это не нарушает общности входной модели.

3.3.6. Выводы

1. Модель одного и того же объекта может быть построена различными способами. Возможность выбора подразумевает рациональное построение модели с ориентацией на решаемую задачу. (Можно было бы ставить целью оптимизацию модели, но сегодня постановка такой задачи кажется слишком сложной).

2. При построении модели следует руководствоваться следующей схемой ее использования:

получение и загрузка данных → внешняя модель → обработка и хранение данных ← переход к машинной модели ←

← внутренняя модель ← обработка и хранение данных ← решение задачи → переход к внешней модели → выдача результата

Элементы этой схемы характеризуются (с интересующей нас точки зрения) трудоемкостью и используемыми ресурсами (устройства, объем памяти и др.). Рациональный выбор модели заключается в снижении общей трудоемкости при заданных ограничениях на ресурсы.

3. Построение модели- итерационный процесс (как и всякий процесс проектирования). При возникновении противоречия следует искать возможности их разрешения на предыдущих этапах.

### 3.4. Методы решения оптимизационных задач

Оценка сложности решения задачи может быть охарактеризована тремя параметрами: возможностью получения аналитического решения; непрерывностью пространства поиска; учетом случайных факторов. В соответствии с этим множество методов решения по каждому из перечисленных параметров делится на два взаимоисключающие подмножества:

- аналитические - численные;
- непрерывные - дискретные;
- детерминированные - стохастические.

Таким образом, имеем восемь непересекающихся подмножеств методов. По затратам времени на решение самые экономные - аналитические, непрерывные, детерминированные. Переход в противоположное подмножество (численные, дискретные, стохастические) может увеличить время решения в $10^4...10^5$ раз.

По способам получения решения наиболее известны следующие методы решения оптимизационных задач:

- прямые методы;
- линейное программирование;
- дискретное программирование;
- нелинейное программирование;
- динамическое программирование и др.

#### 3.4.1. Прямые методы отыскания экстремума

1. Простой перебор. Метод заключается в последовательной проверке заданных векторов переменных. Выбирается вектор, соответствующий оптимальному значению функции цели. Векторы задают списком (перебор на списке) или получают дискретным разбиением пространства поиска (перебор на решетке). Ограничений на применение метода нет.

2. Пропорциональное деление отрезка (дихотомия, метод золотого сечения и др.). Метод применим для унимодальной на отрезке функции. Заключается в последовательном сокращении длины рассматриваемого отрезка (по оси переменной) по результатам сравнения значений функции цели в трех или четырех точках. Две из проверяемых точек лежат на концах отрезка, остальные получаются делением отрезка (например, при дихотомии - пополам).

3. Случайный поиск. Заключается в последовательной проверке векторов, получаемых с помощью датчика случайных чисел. Ограничений на применение метода нет.

4. Градиентные методы. Применяются для непрерывных, в области поиска, дифференцируемых функций цели. Метод характеризуется направленным движением к экстремуму функции. Каждая последующая координата проверяемой точки вычисляется по формуле

$$x_i^k = x_i^{k-1} \pm \lambda \, \partial f / \partial x_i \, .$$

где $i$ – номер переменной, $k$ – номер шага поиска, $\lambda$ – коэффициент, определяющий длину шага.

Конец поиска определяется одним из следующих соотношений:

$$\partial f / \partial x_i = 0,$$

$$f(x_i^k) - f(x_i^{k-1}) < \varepsilon,$$

где $\varepsilon$ – заданная погрешность вычислений.

В зависимости от характера задачи применяется несколько разновидностей градиентного метода: наискорейший спуск, покоординатный спуск и др.

### 3.4.2. Линейное программирование

Методы линейного программирования применяются для решения задач, в которых и функция цели и ограничения – линейны.

Пример постановки линейной задачи.

Задача о рациональном питании ставится следующим образом:

найти $\min\limits_{x_i} \sum\limits_{i=1}^{n} C_i x_i$ при ограничениях $\sum\limits_{i=1}^{n} a_{ji} x_i \geq b_j$ ,

где $x_i$ – количество продукта $i$ , $C_i$ – стоимость продукта, $a_{ji}$ – содержание полезной составляющей $j$ в продукте $i$ , $b_j$ – потребная норма составляющей $j$ .

Задачи линейного программирования имеют простую геометрическую интерпретацию. Поскольку ограничения – линейные функции, каждое из них может быть представлено гиперплоскостью в пространстве переменных, а все ограничения образуют выпуклый гипермногогранник. Линейность функции цели приводит к тому, что поверхности равного уровня (геометрическое место точек, имеющих одинаковые значения функции цели) – параллельные гиперплоскости. Так как многогранник ограничений выпуклый, экстремальное значение функции цели, принадлежащее области допустимых решений, лежит на гиперплоскости равного уровня, опорной (касательной) к многограннику. Следовательно, решение совпадает с одной (или несколькими) из вершин многогранника ограничений.

Такая простая интерпретация позволяет построить ряд эффективных алгоритмов поиска, основанных на последовательной проверке значений функции цели в вершинах многогранника ограничений.

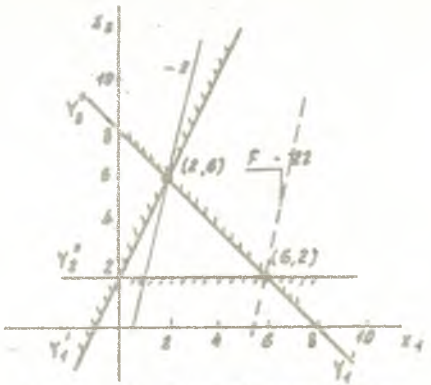
Симплекс-метод решения линейных оптимизационных задач основан на последовательной смене системы координат и направленном движении от вершины к вершине многогранника по ребрам – координатным осям.

Пример.

Найдем минимум функции $F = -4x_1 + x_2$

при ограничениях
$$2x_1 - x_2 + 2 \geq 0;$$
$$-x_1 - x_2 + 8 \geq 0;$$
$$x_2 - 2 \geq 0.$$

Для этого перейдем к косоугольной системе координат ($y_1$, $y_2$, для которой $y_1 = 2x_1 - x_2 + 2$; $y_2 = -x_1 - x_2 + 8$ (см. рис. 10). В новой системе осями координат будут прямые $y_1 = 0$; $y_2 = 0$; начало координат (в старой системе) (2; 6). Функция цели примет вид $F = \frac{1}{5}(-5y_1 + 2y_2 - 6)$, а ее значение в начале координат равно –2. Функция цели убывает при движении в направлении оси $y_1$ ($y_2 = 0$), т.е. при переходе к новым осям сохраняем ось $y_2$ : $y_2 = -x_1 - x_2 + 8$; $y_3 = x_2 - 2$. Функция цели принимает вид $F = 4y_2 + 5y_3 - 22$ и в начале координат (6; 2) равна –22. Дальнейшее уменьшение функции в пределах ограничений невозможно (коэффициенты при переменных положительны).

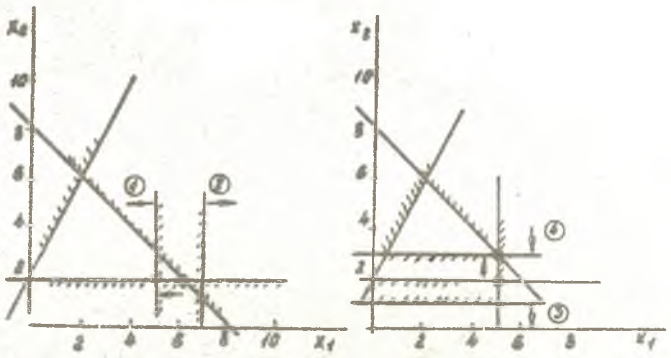



Р и с. 10. Пространство поиска

### 3.4.3. Дискретное программирование

В практике проектирования часто встречаются задачи, в которых область допустимых решений многосвязна (например, ограничения выделяют в пространстве поиска несколько областей). Задачи, в которых переменные принимают только целые значения, - типичные дискретные задачи - на первый взгляд кажутся более простыми, чем соответствующие непрерывные задачи. На самом деле дискретные задачи по трудоемкости на порядок сложнее непрерывных, поскольку поиск глобального экстремума (по всем областям допустимых решений) требует многократного проведения локальной оптимизации и организации выбора.

Эффективный прием дискретного программирования - регуляризация. Прием заключается во временном игнорировании ограничений (всех или части) и последующем переходе от безусловного экстремума в область допустимых решений (рис. II.). Регуляризация позволяет сократить число проверяемых локальных экстремумов, а значит ускорить получение решения.

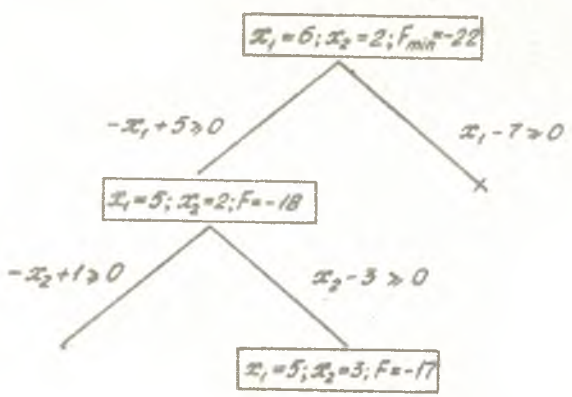


Р и с. II. Последовательное разбиение пространства поиска

Рассмотрим решение дискретных задач на конкретном примере с применением метода ветвей и границ. (Метод ветвей и границ применяют не только для решения дискретных задач.)

Пример. Введем в задачу, рассмотренную в предыдущем примере, дополнительное ограничение : $x_1, x_2$ - целые, нечетные числа.

Используя прием регуляризации, найдем решение в непрерывной области поиска симплекс-методом: $x_1 = 6;\ x_2 = 2;\ F_{min} = -22$ . Полученное решение не удовлетворяет условию дискретности. По переменной $x_1$ ближайшее к полученному решению значения 5 и 7. Разобьем область поиска на две части, введя для каждой по дополнительному ограничению: $-x_1 + 5 \geq 0$ и $x_1 - 7 \geq 0$ (см. рис. II ). Вторая из двух полученных областей - пустое множество (система неравенств не имеет решений). Поиск решения в первой (регулярной) области методами линейного программирования приводит к результату: $x_1 = 5;\ x_2 = 2;\ F = -18$ . Этот результат не удовлетворяет условию дискретности по переменной $x_2$. Разобьем область поиска на две части, введя дополнительные ограничения: $-x_2 + 1 \geq 0$ и $x_2 - 3 \geq 0$. Первая их полученных областей пуста, во второй ищем решение методами линейного программирования: $x_1 = 5;\ x_2 = 3;\ F = -17$ . Полученный результат удовлетворяет условиям дискретности. Анализируя ход решения, можно сделать вывод, что решение дискретной задачи было сведено к многократному повторению непрерывных решений, при этом для организации процесса использовался метод ветвей и границ (рис. I2).




Р и с. I2. Метод ветвей и границ

### 3.4.4. Нелинейное программирование

Класс нелинейных задач очень широк: к нему относятся любые задачи, кроме линейных. Но обычно при рассмотрении методов нелинейного программирования ищут решение детерминированных непрерывных задач, в которых целевая функция или ограничения нелинейны. Методы нелинейного программирования или универсальны, или предназначены для решения какой-либо группы нелинейных задач (например, квадратичных).

Рассмотренный ниже метод штрафных функций позволяет преобразовать оптимизационную задачу с ограничениями в последовательность задач, каждая из которых не имеет ограничений. Вследствие того, что методы штрафных функций не оперирует ограничениями в явном виде, они оказались эффективными в вычислительном отношении для задач нелинейного программирования с нелинейными ограничениями.

Пусть требуется решить задачу $\Phi(x) = \min\limits_{x \in G} f(x)$,

где $G$ – область допустимых решений, заданная ограничениями $\varphi_i(x) \geq 0$.

Общая идея метода заключается в том, что задача отыскания условного экстремума заменяется задачей отыскания безусловного экстремума (при этом ограничения удовлетворяются). Новая задача формулируется в виде

$$F(x) = \min_x \left[ f(x) + \delta_k(x/G) \right],$$

где

$$\lim_{k \to \infty} \delta_k(x/G) = \delta(x/G),$$

$$\delta(x/G) = \begin{cases} 0, & \text{если } x \in G, \\ +\infty, & \text{если } x \notin G; \end{cases}$$

$k$ – номер итерации.

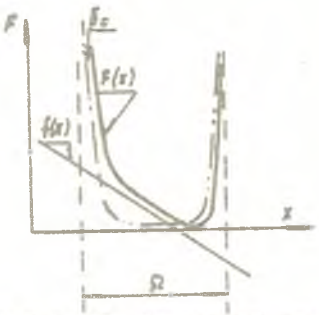
Такая постановка задачи позволяет применить для ее решения простые методы (например, градиентный поиск). Задача решается многократно и на каждой итерации функция $\delta(x/G)$ меняется. При увеличении $k$ задача приближается к исходной, т.е. увеличивается точность решения. Заданная точность решения определяет число итераций.

В практической реализации метода различают внутренний и внешний штрафы. Внутренний штраф (рис. 13) применяется, когда область $G$ задана неравенствами $\varphi_i(x) \geq 0$, где $\varphi_i(x)$ – непрерывные функции, $i = 1, 2, \ldots, m$. Для вычисления штрафа используют функции

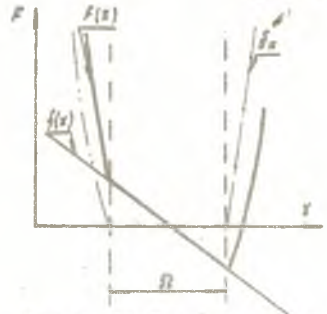
$$\delta_k(x/G) = -r_k \sum_{i=1}^{m} \ln \varphi_i(x);$$

$$\delta_k(x/G) = r_k \sum_{i=1}^{m} 1/\varphi_i(x),$$

где $r_k$ – параметр штрафа (положительная, монотонно убывающая последовательность чисел, сходящаяся к 0, например $r_k = 1, 1/2, \ldots, 1/k$).




Р и с. I3. Внутренний штраф




Р и с. I4. Внешний штраф

Начальная точка поиска принадлежит области $G$ и поиск ведется в области допустимых решений, т.е. поиск ведется "изнутри" области и выход на границу (если это необходимо) осуществляется медленно.

Внешний штраф (рис. I4.) используется, когда область $G$ задана системой равенств и неравенств

$$\varphi_i(x) \le 0, \quad i = 1, 2, \ldots, m;$$
$$\varphi_i(x) = 0, \quad i = m+1, \ldots, l$$

В этом случае решение всегда лежит на границе допустимой области (часть ограничений – точные равенства) и поэтому оптимизируемая функция $F(x)$ выбирается таким образом, чтобы возрастание штрафной составляющей начиналось за границей.

Штраф вычисляется по формулам

$$\delta_k(x/G) = r_k \psi(x);$$

$$\psi(x) = \sum_{i=1}^{m} (\varphi_i^+(x))^2 + \sum_{i=m+1}^{l} (\varphi_i(x))^2 \quad \text{или}$$

$$\psi(x) = \sum_{i=1}^{m} \varphi_i^+(x) + \sum_{i=m+1}^{l} |\varphi_i(x)|;$$

$$\varphi_i^+(x) = \max\{0, \varphi_i(x)\} \quad \text{– срезка } \varphi_i(x).$$

Начальная точка поиска выбирается произвольно, параметр штрафа монотонно возрастает (например, $r_k = 1, 2, \ldots, k$).

### 3.4.5. Динамическое программирование

Динамическое программирование применяется для решения задач динамики полета, управления, экономических и других. Главные условия применимости методов динамического программирования: аддитивность и сепарабельность функции цели (т.е. функция цели представляет собой сумму, причем каждая переменная входит только в одно слагаемое). Следует заметить, что существуют приемы преобразования задач к виду, в котором эти условия выполняются.

Методы динамического программирования основаны на принципе оптимальности Беллмана, который для дискретных процессов может быть сформулирован следующим образом: последующие решения должны быть оптимальными относительно предыдущего состояния, полученного на предыдущем этапе (независимо от того, каково это состояние и как оно получено). Другая формулировка принципа оптимальности: функция цели, принимающая на $k$-м шаге оптимальное значение при $x = x_k$, принимает также оптимальное значение для оставшихся $k-1$ шагов при том же $x = x_k$.

Для непрерывных процессов оптимальную траекторию можно представить как единую цепь, каждый участок-звено которой является экстремалью. Принцип оптимальности Беллмана требует, чтобы только конечный участок оптимальной траектории был оптимален. Это условие определяет применимость принципа оптимальности: если оно несправедливо, то методы динамического программирования не подходят для решения задачи.

Рассмотрим формальный математический аппарат динамического программирования на примере следующей задачи распределения ресурсов.

Пример. Имеется четыре отрасли и $y$ средств, которые надо распределить между ними оптимальным образом. Функции прибыли по отраслям заданы таблицей:

| Вложения в отрасль, млн. руб. | Прибыль по отраслям | | | |
|---|---|---|---|---|
| | $g_1(y)$ | $g_2(y)$ | $g_3(y)$ | $g_4(y)$ |
| I | 0,20 | 0,15 | 0,10 | 0,22 |
| 2 | 0,25 | 0,30 | 0,30 | 0,40 |
| 3 | 0,40 | 0,45 | 0,55 | 0,50 |
| 4 | 0,60 | 0,55 | 0,70 | 0,60 |

Требуется максимизировать суммарный доход

$$I(y) = \max_{u_k} \sum_{k=1}^{4} g_k(u_k); \qquad \sum_{k=1}^{4} u_k = y.$$

Распределение ресурсов условно можно разделить на этапы. Первый этап – вложение всех средств в одну отрасль, второй – в две и так далее. Функциональное уравнение Беллмана в общем случае запишется в виде $S_k(y) = \max_{0 \le u_k \le y} [g_k(y-u_k) + S_{k-1}(u_k)]$.

Конкретно $S_1(y) = g_1(y)$;

$$S_2(y) = \max_{0 \le u_2 \le y} [g_2(y-u_2) + g_1(u_2)];$$

$$S_3(y) = \max_{0 \le u_3 \le y} [g_3(y-u_3) + S_2(u_3);]$$

$$S_4(y) = \max_{0 \le u_4 \le y} [g_4(y-u_4) + S_3(u_4)].$$

Ход решения и результаты представлены следующей таблицей:

| $y$ | $S_1(y)$ | $S_2(y)$ | $S_3(y)$ | $S_4(y)$ | Вложения по отраслям | | |
|---|---|---|---|---|---|---|---|
| | | | | | 2 | 3 | 4 |
| I | 0,20 | 0,20 | 0,20 | 0,22 | I,0 | I,0,0 | 0,0,0,I |
| 2 | 0,25 | 0,35 | 0,35 | 0,42 | I,I | I,I,0 | I,0,0,I |
| 3 | 0,40 | 0,50 | 0,55 | 0,60 | I,2 | 0,0,3 | I,0,0,2 |
| 4 | 0,60 | 0,65 | 0,75 | 0,77 | I,3 | I,0,3 | 0,0,3,I |

Метод динамического программирования дает существенный выигрыш в трудоемкости, поскольку использование результатов предыдущего шага позволяет сократить число просматриваемых вариантов (по сравнению с прямым перебором).

Характерной особенностью многошаговых процессов, которые оптимизируются с помощью динамического программирования, является зависимость текущего состояния только от предыдущего и независимость от более ранних состояний (отсутствие последействия).

### 3.5. Сложности при решении оптимизационных задач

Многомерность. Проектные задачи как правило многопараметричны. Предположим, что функция цели зависит от одного параметра. Тогда для отыскания ее экстремума выделим в области допустимых значений

параметра $n$ точек и проверим значения функции цели в этих точках. В том случае, когда функция цели зависит от двух переменных, для проверки области допустимых решений с той же точностью потребуется $n^2$ испытаний. Если в задаче $m$ переменных, то число испытаний возрастет до $n^m$. Проведение каждого испытания потребует времени, и с ростом числа переменных общее время поиска экстремума может оказаться настолько большим, что решение задачи потеряет смысл.

Многоэкстремальность. Часто функция цели имеет не один, а несколько экстремумов. В этом случае ее решение разбивается на два этапа: многократный поиск локальных экстремумов, выбор глобального экстремума из множества локальных. Очевидно, что сложность (а следовательно и время решения) такой задачи увеличивается по сравнению с одноэкстремальной (имеющей унимодальную функцию цели).

Дискретность. Иногда область допустимых решений, заданная ограничениями, становится многосвязной. Разделение области поиска на несколько частей требует многократного повторения процесса решения, что приводит к увеличению трудоемкости. Частный случай дискретной задачи – требование целочисленности решения.

Особенности функции цели. В некоторых случаях решение усложняется в связи со сложным характером функции цели или ограничений. Примером может служить овражная функция, поиск экстремума которой требует особых подходов.

Перечисленные сложности можно преодолеть двумя путями: изменением формулировки задачи (как на этапе предварительной постановки, так и на этапе формализации); выбором соответствующего метода решения задачи.

### 3.6. Машинная графика

#### 3.6.1. Основные понятия

Машинная графика – необходимый и наиболее эффективный элемент автоматизированного проектирования. Подробное изучение подходов и методов, применяемых в машинной графике, не входит в задачу данного курса, не знакомясь с "Основами САПР", необходимо составить некоторое общее представление о месте этого раздела. Считается, что человек более эффективно обрабатывает графические объекты, чем ЭВМ. В проектировании технических объектов работа с изображениями занимает очень много места и часто не требует высокой квалификации. Передача ЭВМ простых, но трудоемких операций с графическими объектами разгру-

жает конструктора и вместе с одновременным сокращением информационных потоков дает значительный выигрыш в сроках проектирования. В некоторых случаях естественные пределы восприятия не позволяют человеку обрабатывать сложные изображения, и тогда применение ЭВМ неизбежно.

Машинная графика охватывает круг задач, связанных с генерацией, представлением, обработкой или оценкой графических объектов вычислительной машиной, манипулированием графическими объектами и установлением связи между ними и неграфической информацией. Машинная графика включает: структуры данных, графические алгоритмы и программы, языки (т.е. захватывает части информационного, математического, программного и лингвистического обеспечения). К объектам машинной графики относятся: символы, линии, тела, области, фотографии.

Различают три группы задач: изобразительная графика (построение и преобразование моделей и изображений, идентификация объектов); анализ изображений (распознавание образов, оценка и повышение качества изображений); анализ сцен (распознавание изображений и их связей).

Элементы данных в машинной графике:

к о о р д и н а т н а я  т о ч к а – совокупность и значения координат;

г р а ф и ч е с к и й  п р и м и т и в – элемент, генерируемый процессором;

г р а ф и ч е с к и й  о б ъ е к т – множество примитивов, идентифицируемых одним именем.

Данные, описывающие объект, можно разделить на две части: инициализирующие (имя, начальная точка, параметры); специфицирующие (описания примитивов).

Методы построения и анализа изображений используют математический аппарат аналитической геометрии, начертательной геометрии, теории множеств, математической логики, исследования операций и других разделов математики.

### 3.6.2. Применение машинной графики в САПР

Создание удобного графического инструмента САПР состоит в разработке геометрических моделей, программ загрузки данных, алгоритмов и программ формирования чертежа (в интерактивном или автоматическом режимах), программ вычерчивания (на экране или графопостроителе).

В последнее время широкое распространение получили системы компьютерного проектирования (*CAD*). Основное назначение этих систем – графическая поддержка конструктора. Но помимо этих функций системы выполняют также ряд несложных вычислений, связанных с конструированием. Для представления о их возможностях рассмотрим краткое описание системы *AutoCAD* (фирма *AUTODECK*). Система может изготавливать разнообразные чертежи и схемы. Для управления работой системы предусмотрен набор команд типа меню. Использованы следующие графические элементы:

- точки и прямые линии различной толщины;
- кривые произвольной толщины (последовательности сегментов);
- окружности, дуги;
- текстовые фрагменты произвольного размера;
- объемные призматические примитивы (элементы трехмерных изображений);
- графические примитивы, внесенные пользователем в библиотеку системы;
- изображения, полученные ранее (при проектировании).

В системе используются световое перо, электронный планшет, сенсорная панель, клавиатура. Точки на экране задаются позицией курсора. Их можно использовать для ввода координат или для определения элементов изображения (прямых, окружностей и т.д.). Для проработки деталей чертежа используются панорамирование и увеличение фрагментов изображения. Фрагменты задаются противолежащими углами. При панорамировании в окно на экране выводят соседний фрагмент того же размера. Возможно перемещение деталей изображения с помощью сетки на расстояние, кратное ее шагу. Изображение, полученное на экране, можно целиком или частично вывести на графопостроитель.

Система имеет вспомогательные средства, позволяющие вычислять расстояния между точками и площади фигур.

*AutoCAD* строится на базе персонального компьютера *IBM PC* или *IBM PC-XT* с объемом памяти на менее 256 К.

## 4. ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ

### 4.I. Последовательность разработки программ

По характеру выполнения работ процесс создания программного изделия подразделяется на ряд последовательных этапов:

- постановка задачи и разработка технического задания (ТЗ);
- составление проекта программы;
- алгоритмизация;
- программирование;
- препарация и трансляция;
- отладка программы;
- оформление программы;
- эксплуатация программы;
- сопровождение программы.

Необходимость исправления ошибок, выявленных на этапах отладки и эксплуатации программы, приводит в возвратам и повторению этапов трансляции, программирования и алгоритмизации. Возможно также внесение изменений в проект при сопровождении программ с последующим повторением всех этапов. Примерное статистическое распределение времени на выполнение отдельных этапов (при условии, что сумма времени на первые семь этапов - I00%) выглядит следующим образом:

I5% - техническое задание и проект;
I5% - алгоритмизация;
I0% - программирование, препарация и трансляция;
50% - отладка;
I0% - оформление.

Начинающие программисты часто недооценивают трудоемкость этапов проектирования, отладки и оформления. Учет приведенных данных при планировании работы способствует успешной разработке программы.

Составление ТЗ имеет следующие цели: уяснение постановки задачи, формулировка требований к программе, их оценка, обсуждение и последующая корректировка. ТЗ содержит следующие сведения: назначение программы, обоснование разработки, требования, этапы и сроки. Составление проекта программы включает разработку структур данных, определение методов решения задачи, выделение функциональ-

ных частей, установку взаимосвязей между ними, формулировку требований к обмену.

А л г о р и т м, по определению, -"точное предписание, определяющее вычислительный процесс, ведущий от варьируемых начальных данных к искомому результату". Основные свойства алгоритма:

- Конечность (работа алгоритма должна заканчиваться за конечное число шагов);
- определенность (трактовка любого предписания должна быть однозначной и доступной для исполнителя);
- массовость (решение должно обеспечиваться для широкого класса задач, отличающихся входными данными);
- результативность (при любом наборе данных алгоритм должен давать некоторый результат);
- эффективность (применение алгоритма должно давать результат за конечное ограниченное время).

При разработке алгоритма необходимо учитывать ресурсы ЭВМ и возможности операционной системы. Для записи алгоритма используют схемы, общепринятые математические средства, элементы алгоритмических языков, словесные формулировки.

П р о г р а м м а, согласно ГОСТ 19.781-74, определяется как "алгоритм, записанный в форме, воспринимаемой вычислительной машиной". Уместно привести и другие определения программы: "последовательность элементарных шагов, доступная для распознавания и выполнения ЭВМ и обеспечивающая получение результата"; "совокупность преобразований и связей, соотносящих структуры данных и носителей информации" [12]. Даже рассмотрение приведенных определений показывает, что программирование это не простой перевод алгоритма (кодирование), а творческий процесс.

Препарация программы состоит в ее переносе на машинные носители, получении распечатки и сверке последней с исходным текстом. Трансляция программы осуществляется средствами операционной системы и заключается в подготовке программы к работе. Одновременно осуществляется поиск синтаксических ошибок.

Процесс отладки направлен на проверку работоспособности программы на всех режимах и заключается в обнаружении, локализации и устранении ошибок в программе.

Разработка программы считается законченной после завершения отладки и оформления программных документов согласно ГОСТ 19.101-77. С передачей программы в эксплуатацию работа над ней не

заканчивается. Процесс сопровождения программного изделия заключается в модификации, обусловленной необходимостью устранения выявленных при эксплуатации ошибок и изменения функциональных возможностей программы.

### 4.2. Оценка качества программ

Анализ программ на современном уровне технологии программирования позволяет ранжировать критерии качества в следующем порядке. Хорошая программа:

- работает согласно ТЗ и это легко проверить;
- широко используется и доступна;
- надежна и приспособлена к выявлению ошибок;
- эффективна по быстродействию и памяти;
- быстро разработана и отлажена.

Работоспособность - очевидное качество программы. Но иногда между разработчиком и заказчиком возникают противоречия в оценке функционирования программы, причем эти противоречия могут выявиться в процессе эксплуатации. Эталоном для оценки работы программы должно быть ТЗ на ее разработку. Широта использования программы зависит не только от ее универсальности, но и от удобства работы с ней, приспособленности к модификации. Под надежностью программы понимается ее работоспособность во всем диапазоне входных данных и на всех режимах. Иерархия качеств относит быстродействие и объем памяти программы на одно из последних мест (начинающие программисты часто считают эти качества самыми важными и уделяют им наибольшее внимание). В тех случаях, когда от создания программы зависит выполнение других важных и срочных работ, скорость разработки и отладки, как и критерий качества выступает на одно из первых мест.

### 4.3. Проектирование программ

Выбор общей стратегии проектирования программ позволяет правильно построить работу и распределить функции разработчиков. Наиболее распространены следующие стратегии: хаотическое, восходящее и нисходящее проектирование, метод расширения ядра.

Применение этих методов подразумевает разделение программы на блоки (модульное программирование). При детальной разработке проекта используется ряд стандартных приемов и методов, в частности структурное программирование.

1. Восходящее проектирование. Иногда возникает необходимость создания программы на базе элементарных готовых подпрограмм. В этом случае разработка ведется "снизу вверх" путем надстройки программ более высокого уровня. Другой случай использования восходящего проектирования предусматривает проверку на нижнем уровне сомнительного метода решения и, в случае успешной работы, развитие программы "вверх".

В общем случае метод заключается в следующем:

- определяется набор базовых функций;
- разрабатываются и отлаживаются программные модули, реализующие базовые функции;
- разрабатываются и отлаживаются подпрограммы, использующие модули нижнего уровня;
- процесс повторяется вплоть до программы верхнего уровня.

Недостатком метода является сложность отладки при объединении модулей.

2. Нисходящее проектирование программ. Разработка программ "сверху вниз" основана на последовательной декомпозиции задач. В процессе разработки проводится пошаговое уточнение с нарастающей детализацией. На каждом шаге планируется выполнение основных функций, соответствующих выбранному уровню. Процесс продолжается до тех пор, пока функции не станут настолько простыми, что каждой из них будет соответствовать один программный модуль. Таким образом, получается иерархическая система программных модулей. При проектировании "сверху вниз" возможна принципиальная нереализуемость некоторых функций модулями нижних уровней, что потребует повторения процесса проектирования. Несмотря на этот недостаток, метод широко применяется, поскольку считается наилучшим с точки зрения затрат на проектирование. Метод позволяет расширить фронт работ вследствие подключения разных программистов для разработки нижних уровней, но при этом администратор должен исключить дублирование отдельных функций внутри различных модулей.

3. Метод расширения ядра. Метод применяется при проектировании программ, использующих сложные структуры данных. В этом случае функции преобразования данных (копирование, сортировка, ввод, вывод) составляют значительную часть программы. Проектирование программы ведется по двум направлениям: определяются составные части программы, устанавливаются межмодульные управляющие связи,

реализующие схему обработки данных. Структура программы определяется не только методом решения, но и физическим представлением данных, а также готовым набором (ядром) универсальных модулей.

4. Модульное программирование. Любой из перечисленных выше методов проектирования основан на том, что программа состоит из фрагментов (модулей), причем каждый модуль разрабатывается отдельно. При создании модуля следует стремиться к его широкому использованию и удобству работы с ним в составе большой программы. Эти цели определяют следующие требования к модулю: монофункциональность, независимость, небольшой размер. Модуль реализует одну функцию и имеет один вход и один выход, которые точно определены. Это позволяет использовать его в программах, не вникая в вопросы внутреннего строения и работы самого модуля. Необходимо, чтобы работа модуля не зависела от источника данных, предыстории процесса, способа использования выходных данных. Размер модуля строго не определен: разные авторы рекомендуют от 25 до 100 операторов.

5. Детализация проекта программы. Предварительное описание программы проводится на выбранном уровне проектирования. При этом необходимо отразить четыре раздела: входную информацию, обработку, выходную информацию, ограничения. Подробность описания определяется возможностью дальнейшего разбиения программы на модули.

Исходную информацию для разработки структуры программы дает описание обработки. Определенные коррективы в структуру могут внести ограничения на быстродействие и объем памяти. Выбор структуры программы - это поиск компромисса между высокой степенью модульности и минимизацией связей внутри программы. Увеличение числа модулей уменьшает время разработки, отладки и сопровождения программы. Неизбежный при увеличении модульности рост числа связей снижает быстродействие и надежность программы, вызван необходимость дополнительных пересылок данных. Опыт показывает, что наиболее предпочтительны иерархическое и последовательно-модульное строение программ (рис. 15).

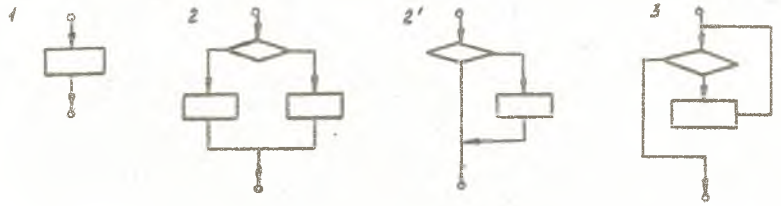
Разработка внутреннего обмена данными включает:

- анализ входной и выходной информации и выбор логической организации данных;
- выбор абстрактных структур данных;
- выбор физической организации данных;
- выбор взаимодействия модулем.


Р и с. I5.Структуры программ: I – монолитная; 2 – монолитно-модульная; 3 – последовательно-модульная; 4 – иерархическая; 5 – иерархически-хаотическая; 6 – модульно-хаотическая

Логическая организация данных определяется предметной областью решения задачи и выбранными моделями. Абстрактные структуры данных учитывают ограничения (на типы и структуры данных), связанные с языком программирования. Физическая организация данных отражает особенности хранения и обработки информации на машинном уровне.

В последнее время среди программистов широкое распространение получила концепция структурного программирования [5], удовлетворяющая принципу экономии усилий. Ее суть заключается в следующем. Любую правильную программу с одним входом и одним выходом можно записать с использованием только трех логических структур: последовательность операторов; выбор по условию одного из двух операторов; повторение оператора до выполнения условия (рис. I6).




Р и с. I6. Стандартные структурные элементы:
I – безусловный оператор или блок; 2, 2′ – условный оператор; 3 – оператор цикла

Под оператором в данном случае следует понимать как одиночные операторы, так и группы операторов или перечисленных структур. Применяя итерации и вложения этих структур, можно получить программу любого размера и сложности. Структурное программирование должно применяться на всех уровнях написания программы. При этом повышаются доступность программы и производительность программиста. Основные приемы структурирования программы - дублирование фрагментов программы и введение булевого признака или переменной состояния.

## 4.4. Отладка программ

Возникновение ошибок в программе возможно на всех этапах ее разработки. Неполный учет ограничений; перерывы в работе, приводящие к использованию одинаковых имен для различных переменных (или наоборот); нечеткое представление о входных данных; перенос программы и ряд других причин служат источником ошибок. Наличие ошибок в только что разработанной программе - закономерное явление, а уверенность программиста в безошибочности его программы затягивает отладку на неопределенный срок и является вредной.

Подготовка к отладке начинается с этапа ее проектирования и строится по следующей схеме:

- составление общего плана отладки и системы контрольных примеров;
- проверка алгоритмов, выбор контролируемых мест и средств отладки;
- проверка программы, включение в нее средств отладки, изготовление эталонных результатов тестов;
- печать и сверка текста программы, получение и проверка вспомогательных таблиц.

При отладке программист хочет убедиться в том, что программа работает согласно техническому заданию. Предполагается, что программа содержит ошибку, если ее поведение (или результат) отличается от ожидаемого. В процессе отладки многократно выполняются три последовательные операции: контроль программы и результата, локализация ошибки, исправление программы.

Поиск ошибок в программе начинают еще до ввода ее в ЭВМ. При этом используют п р о с м о т р (беглое прочтение текста), п р о в е р к у (мысленное восстановление вычислительного процесса), п р о к р у т к у (имитацию действий ЭВМ в соответствии с программой).

Часть ошибок выявляется при трансляции с помощью системных программ. После запуска программы возможны следующие варианты развития событий:

- программа работает и выдает результат (правильный или неправильный);
- программа начинает работать и аварийно останавливается;
- программа работает без остановки (зацикливается).

Поведение программы дает информацию для определения характера ошибки и ее локализации. Правильность результата определяется сравнением его с эталонным. Информация, включающая набор специально подобранных данных и соответствующий им эталонный результат, называется т е с т о м. Существует ряд правил правильного и экономного составления тестов (одним тестом проверить программу невозможно).

Установление точного места ошибки в программе проводится в следующей последовательности: предположение о характере и месте ошибки, проверка предположения по тексту, получение дополнительной информации при повторном запуске программы.

Источниками дополнительной информации могут быть: аварийная печать (осуществляется средствами операционной системы при аварийной остановке), печать в узлах (планируется программистом), специальные отладочные программы.

Исправление программы - ответственная операция, требующая особого внимания, поскольку любое изменение может послужить источником новых ошибок.

## 4.5. Документирование программ

Широкое использование программ, их эксплуатация и сопровождение невозможны без доступной документации.

Разработанная в нашей стране Единая система программной документации (ЕСПД) упрощает и облегчает документирование программ. В стандартах ЕСПД установлены требования, регламентирующие разработку, производство, тиражирование и сопровождение программ. Стандарты распространяются на программы и соответствующую документацию независимо от их назначения и области применения. Обозначения стандартов строятся по классификационному признаку следующим образом:

Разбиение стандартов ЕСПД на группы: 0 - общие положения; 1 - основополагающие стандарты; 2 - правила выполнения документации разработки; 3 - правила выполнения документации изготовления; 4 - правила выполнения документации сопровождения; 5 - правила выполнения эксплуатационной документации; 6 - правила обращения документации; 7 и 8 - резервные группы; 9 - прочие стандарты.

Виды программных документов и программы определяет ГОСТ 19.101-77. Программы делятся на компоненты (выполняющие законченную функцию) и комплексы (состоящие из двух и более компонентов). Виды программных документов: спецификация (определяет состав программы и документации); ведомость держателей подлинников (перечень предприятий, на которых хранятся подлинники программных документов); текст программы; описание программы (сведения о структуре и функционировании); программа и методика испытаний (требования, подлежащие проверке; порядок и методы контроля); техническое задание (назначение, область применения, требования, стадии и сроки разработки); пояснительная записка (схема и общее описание алгоритма, обоснование решений); эксплуатационные документы (сведения для обеспечения функционирования и эксплуатации). Оформление программных документов соответствует правилам оформления текстовых документов.

В практике разработки программ применяются и другие методы документирования (например, метод ХИПО, основанный на последовательной многоуровневой записи проекта программы по схеме "вход-обработка-выход" при нисходящей разработке).

## 5. ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ САПР

### 5.1. Основные понятия и определения

Информационное обеспечение САПР включает документы, содержащие описания стандартных проектных процедур, типовых проектных решений и данные, а также файлы и блоки данных на машинных носителях с записью этих документов. Составная часть информационного обеспечения - банки данных.

Банк данных - это автоматизированное хранилище информации, включающее базы данных (БД) и систем управления базами данных (представляющие собой набор управляющих программ). База данных -

это совокупность взаимосвязанных, хранящихся вместе данных. К БД предъявляют следующие требования: минимальная избыточность (повторная запись одной и той же информации, вызванная удобством идентификации, поиска и обработки данных, должна сводиться к минимуму); логическая и физическая независимость (файлы, записи и элементы данных должны представлять собой логически завершенные конструкции, а их размещение в памяти ЭВМ должно быть упорядочено; очевидно это требование вступает в противоречие с первым); возможность и необходимость управления. Системы управления БД обеспечивают пользователя инструментом для оперирования данными (поиск, модификация, включение, удаление) и выполняют ряд дополнительных функций (обеспечение секретности, защита целостности, синхронизация, защита от отказов и восстановление).

Ф а й л-это совокупность данных, состоящая из логических записей, относящихся к одной теме. З а п и с ь-это набор полей, объединенных логикой использования в программе. П о л е (э л е м е н т  д а н н ы х) - законченная по смыслу порция данных:


Информационный объект состоит из характеристик: символов, чисел, кодов. Элемент включает характеристику объекта и может выступать в роли а т р и б у т а или к л ю ч а. Ключ дает возможность идентифицировать запись (отличить ее от остальных).

### 5.2. Уровни абстракции в представлении данных

При проектировании БД выделяют четыре уровня абстракции:

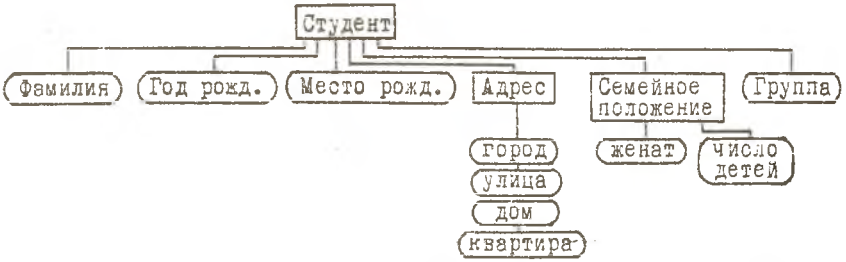
внешнее представление-совокупность требований к данным;

концептуальная модель - представление совокупности элементов данных и их связей для хранения в БД и обработки;

логическая модель - структурная схема данных;

физическая модель - схема размещения данных на устройствах памяти.

В качестве концептуальной модели часто используют ориентированный граф (рис I7.)

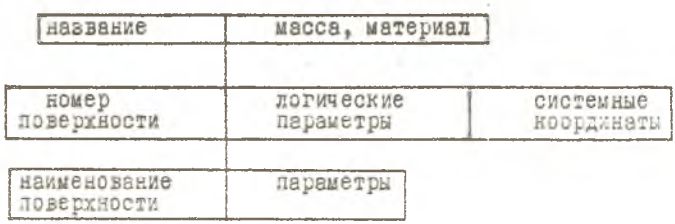



Р и с.I7. Концептуальная модель "Информация о студенте"

Построение логической модели имеет следующие цели:

- определение ключей и атрибутов;
- выявление и классификация связей;
- устранение избыточности;
- построение иерархий.

Связи между элементами классифицируются по признакам: один к одному; один к многим (многие к одному); многие к многим. Пример логической структуры данных (см. разд. 3.3.5.) приведен на рис. I8.




Р и с. I8. Логическая схема записи "Деталь"

Основная проблема в физическом представлении БД - это хранение файла. Ф о р м а т з а п и с и представляет собой список имен полей, причем каждое поле занимает фиксированное число байтов и имеет фиксированный тип данных. Файловые системы, использующие дисковую память, обычно разделяют на физические блоки равного размера (от $2^9$ до $2^{12}$ байт). Каждый блок имеет адрес. Файл хранится в одном или более блоках, в каждом блоке могут храниться несколько записей. Запись имеет адрес (либо абсолютный адрес первого байта, либо адрес блока и смещение).

## 5.3. Модели данных

### 5.3.1. Реляционная модель данных

В основе этой модели лежит математическое понятие теоретико-множественного о т н о ш е н и я, которое представляет собой подмножество декартова произведения списка доменов. Д о м е н - это просто множество значений. Элементы отношения называются к о р т е ж а м и.

Пример. Имеем два домена $D_1 = \{0, 1\}$ и $D_2 = \{a, b, c\}$. Тогда произведение этих доменов образует множество кортежей $\{(0, a), (0, b), (0, c), (1, a), (1, b), (1, c)\}$.

Удобно представлять отношение как таблицу, где каждая строка есть кортеж и каждый столбец соответствует одному компоненту. Столбцы называются при этом а т р и б у т а м и, и им присваивают имена.

Пример. В таблице "Ведомость успеваемости" представлено отношение с атрибутами ФАМИЛИЯ, ФИЗИКА, МАТЕМАТИКА.

| ФАМИЛИЯ | ФИЗИКА | МАТЕМАТИКА |
|---|---|---|
| Абрамов А.А. | отлично | хорошо |
| Борисов Б.Б. | хорошо | отлично |
| Васильев В.В. | отлично | отлично |

Кортежем является, например, элемент (Абрамов А.А., отлично, хорошо).

Реляционная модель данных оперирует только с одной конструкцией, которую должен понимать пользователь, поэтому она считается лучшей по критерию легкости использования. Недостатком модели является ее низкая эффективность (большой объем памяти, трудность реализации отношений "многие к многим").

### 5.3.2. Сетевая модель данных

Сетевая модель данных - это модель объектов-связей, где допускаются только бинарные связи "многие к одному". Представляется в виде ориентированного графа (с е т и). Реализация связей упрощается (по сравнению с реляционной моделью).

Отношения в этой модели представляют л о г и ч е с к о й з а п и с ь ю с компонентами - полями.

Пример. Данные представлены графом (сетью) согласно рис. I9.


Р и с. I9. Сетевая схема данных

Возможна реализация отношений:
КАФЕДРА ведет занятия в ГРУППАХ КАФЕДРА_ГРУППЫ
КАФЕДРА преподает ДИСЦИПЛИНЫ КАФЕДРА_ДИСЦИПЛИНЫ и т.д.
Возможны логические записи вида:
КАФЕДРА(НАЗВАНИЕ,ЧИСЛО ПРЕПОДАВАТЕЛЕЙ).

5.3.3. Иерархические модели данных

В том случае, когда сеть представляет собой совокупность деревьев (имеется в виду модель в виде графа), мы имеем дело с иерархической моделью. Примером иерархической модели может служить схема, представленная на рис. I8. Иерархические модели считаются наиболее эффективными по использованию памяти, поэтому сетевые структуры часто приводят к иерархическим.

5.4. Базы знаний

Развитие САПР привело к появлению э к с п е р т н ы х с и с т е м, одной из составных частей которых являются б а з ы з н а н и й (БЗ). Экспертные системы предназначены для решения нестандартных задач синтеза в диалоговом режиме. БЗ наряду с данными (фактами) содержат информацию о предметной области, о языке общения с пользователем, о структуре и принципах работы экспертной системы. Эта информация записывается в виде п р а в и л, с е м а н т и ч е с к и х с е т е й, ф р е й м о в.

Правила представляют собой логические предложения вида "ЕСЛИ (сопоставление) ТО (выполнение)".

Семантические сети - это запись отношений между информационными элементами, которая структурно аналогична описанным выше сетевым моделям данных, но реализуется на языках высокого уровня (ПРОЛОГ, ЛИСП).

Фреймы представляют собой шаблоны, автоматическое заполнение которых в процессе работы с экспертной системой приводит к генера-

ции программ. Для заполнения фрейма используются факты, правила, программные модули.

Примерная структура данных для представления фрейма имеет вид:

<фрейм>::= <имя> { <условие применимости>|... } { <ссылка на прототип>|...} {<предок>|...} {<потомок>|...} <описатель фрейма>|... | <описатель фрейма> <слот>|...|<слот> { <терминал фрейма>|... } ,

где < > - ограничители структурного элемента;

::= - это есть;

{ } - ограничители необязательных элементов структуры;

| - разделители одинаковых элементов;

Слот - это фрагмент структуры фрейма вида

<слот>::= <имя слота> <порядковый номер слота>{<описатель типа> <указатель>|... } <правило заполнения> <тело слота> {<комментарий>|... } ,

где <тело слота>::=<значение>|... |<значение> {<процедура>|... ... } {<ограничение>|...} {<терминал слота>|... },

<терминал слота>::= <значение> .

В процессе интерпретации запроса и вывода решения в конкретный экземпляр фрейма включают необходимые значения слотов и данные терминалов.

В общем случае фреймы представляют собой совокупность описаний и связанных с ними процедур. Совокупность фреймов в БЗ представлена фреймовой сетью в виде ориентированного двудольного графа, где множество вершин-фреймов соединены дугами с множеством вершин-слотов (те и другие заданы своими именами).

Системы представления знаний включают набор фреймов и программных средств, обеспечивающих создание, ведение и коррекцию БЗ.

## Библиографический список


I. Безбородов Ю.М. Индивидуальная отладка програмы. М.: Наука, 1982. 192 с.

2. Гаврилов В.Н. Автоматизированная компоновка приборных отсеков летательных аппаратов. М.: Машиностроение, 1988. 136 с.

3. Прикладное математическое обеспечение САПР. Методы проектирования и документирования программ: Учеб. пособие/ А.И. Данилин; Куйбышев. авиац. ин-т. Куйбышев, 1983. 52 с.

4. Единая система программной документации (ЕСПД) - М.: Госстандарт, 1982.

5. Заглер К. Методы проектирования программных систем. М.: Мир, 1985. 326 с.

6. Зозулевич Д.М. Машинная графика в автоматизированном проектировании. М.: Машиностроение, 1976. 237 с.

7. Кузин Л.Т. Основы кибернетики. М.: Энергия, 1973. 502 с.

8. Лазарев И.Б. Математические методы оптимального проектирования конструкций: Учеб. пособие. Новосибирск:[НИИЖТ], 1974, 187 с.

9. Системы автоматизированного проектирования: Учеб. пособие для втузов/ Под ред. И.П. Норенкова. М.: Высш. школа, 1986. Т. 1-9.

10. Ступаченко А.А. САПР технологических операций. Л.: Машиностроение. Ленингр. отд-ние, 1988. 234 с., ил.

11. Ульман Дж. Основы систем баз данных / Пер. с англ. М.Р. Когаловского и Б.В. Когутовского; Под ред. М.Р. Когаловского. М.: Финансы и статистика, 1983. 334 с., ил.

12. Йодан Э. Структурное проектирование и конструирование программ /Перевод с англ.; Под ред. Л.Н.Королева. М.: Мир, 1979. 416 с.

ОГЛАВЛЕНИЕ


1. Проектирование и автоматизация..... 3
1.1. Обоснование САПР..... 3
1.2. Требования к САПР..... 4
1.3. Процесс проектирования..... 5
1.4. Основы системного подхода..... 8
1.5. Принципы проектирования..... 9
1.6. Структура САПР..... 9
2. Технические средства САПР..... 10
2.1. Структура ЭВМ..... 10
2.2. Память ЭВМ..... 11
2.2.1. Количество информации..... 11
2.2.2. Устройство памяти..... 12
2.2.3. Кодирование информации..... 12
2.2.4. Запись данных в памяти ЭВМ..... 12
2.3. Центральный процессор ЭВМ..... 14
2.3.1. Команда и ее запись..... 14
2.3.2. Работа процессора..... 14
2.4. Принципы вычислений на ЭВМ..... 15
2.4.1. Программное управление..... 15
2.4.2. Дискретное представление и преобразование информации..... 15
2.4.3. Адресный принцип идентификации данных..... 16
2.4.4. Цикличность обработки информации..... 16
2.4.5. Формализм записи и обработки данных..... 16
2.4.6. Использование обратных связей..... 17
2.5. Характеристики ЭВМ..... 17

3. Математическое обеспечение САПР..... 17
3.1. Этапы решения задачи на ЭВМ..... 17
3.2. Постановка задачи..... 18
3.2.1. Формальная постановка задачи..... 18
3.2.2. Критерий и функция цели..... 18
3.2.3. Многокритериальность..... 19
3.3. Математические модели..... 20
3.3.1. Общие сведения..... 20
3.3.2. Примеры построения моделей геометрического тела..... 21
3.3.3. Процесс простроения модели..... 21
3.3.4. Кусочно-аналитическая модель тела..... 21
3.3.5. Алгебрологическая модель геометрического тела.....24
3.3.6. Выводы..... 28
3.4. Методы решения оптимизационных задач..... 29
3.4.1. Прямые методы отыскания экстремума..... 29
3.4.2. Линейное программирование..... 30
3.4.3. Дискретное программирование..... 32
3.4.4. Нелинейное программирование..... 34
3.4.5. Динамическое программирование..... 36
3.5. Сложности при решении оптимизационных задач..... 37
3.6. Машинная графика..... 38
3.6.1. Основные понятия..... 38
3.6.2. Применение машинной графики в САПР..... 39
4. Программное обеспечение..... 41
4.1. Последовательность разработки программ..... 41
4.2. Оценка качества программ..... 43
4.3. Проектирование программ..... 43
4.4. Отладка программ..... 47
4.5. Документирование программ..... 48
5. Информационное обеспечение САПР..... 49
5.1. Основные понятия и определения ..... 49
5.2. Уровни абстракции в представлении данных.... 50
5.3. Модели данных..... 52
5.3.1. Реляционная модель данных..... 52

5.3.2. Сетевая модель данных.....52
5.3.3. Иерархические модели данных.....53
5.4. Базы знаний.....53
Библиографический список...55

Г а в р и л о в Валерий Николаевич

ОСНОВЫ САПР

Редактор Л.Я.Ч е г о д а е в а
Техн.редактор Г.А.У с а ч е в а
Корректор Н.С.К у п р и я н о в а

Лицензия ЛР № 020301 от 28.II.91

Подписано в печать 14.12.93. Формат 60x84 1/16
Бумага офсетная. Печать оперативная.
Усл.п.л. 3,49. Усл.кр.-отт. 3,73. Уч.-изд.л. 3,8.
Тираж 500 экз. Заказ 197. Арт. С - 12/94.

Самарский государственный аэрокосмический
университет имени академика С.П.Королева.
443080 Самара, Московское шоссе, 34.

---

ИПО Самарского аэрокосмического университета.
443001 Самара, ул. Ульяновская, 18.