

Раздел 1. Интерфейс "человек-компьютер". Основные понятия

О главном термине курса:

интерфейс — в вычислительной технике, система унифицированных связей, посредством которых устройства компьютера взаимодействуют между собой, а также набор средств диалога, взаимодействия между пользователем и компьютером. С помощью интерфейса пользователь управляет работой компьютера: выдаёт задания, отвечает на его запросы, получает информацию о работе программы.

интерфейс — Совокупность средств и правил, обеспечивающих взаимодействие устройств вычислительной машины или системы обработки информации и (или) программ. [ГОСТ 15971 90] [ГОСТ Р 50304 92].

интерфейс — в широком смысле определенная стандартами граница между взаимодействующими независимыми объектами. Интерфейс задает параметры, процедуры и характеристики взаимодействия объектов.

// Словари и энциклопедии на Академике

https://dic.academic.ru/dic.nsf/enc_tech/471/%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81 (05.02.2018.).

Интерфейс человек-компьютер — "...Интерфейс человек-компьютер (ИЧК) (human/computer interface (HCI)): граница, через которую реализуется физическое взаимодействие человека с прикладной платформой..."

// Словари и энциклопедии на Академике

https://official.academic.ru/8262/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D1%87%D0%B5%D0%BB%D0%BE%D0%B2%D0%B5%D0%BA-%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80 (05.02.2018.).

Немного об истории внедрения ЭВМ.

Прикладное программное обеспечение; взаимодействие пользователя с компьютером; интеллектуализация ЭВМ

Традиционно основное внимание уделялось *точности результатов* работы вычислительных систем (ВС). Гораздо меньше внимания уделялось проблеме *удобства* работы с ВС.

60-е.-70-е гг. – ЭВМ окружена кондиционерами, обслуживается десятками инженеров, техников и операторов. Единственный режим использования ЭВМ – *пакетный*.

Посредник во взаимодействии пользователя с ЭВМ – оператор.

УСЛОВИЯ РАБОТЫ «ДИКТУЕТ» МАШИНА.

Решение задач на ЭВМ (основные этапы):

Содержательная постановка задачи	Формальная постановка задачи	Разработка алгоритма	Написание программы	Получение программы на маш.яз.	Отладка, тестирование	Эксплуатация	Анализ результатов
----------------------------------	------------------------------	----------------------	---------------------	--------------------------------	-----------------------	--------------	--------------------

Программное обеспечение (ПО) – комплекс программ, обеспечивающих обработку или передачу данных и предназначенных для многократного использования и применения разными пользователями (часто считают, что в состав ПО следует включать программную документацию). По видам выполняемых функций программное обеспечение подразделяется на: *системное*, *прикладное* и *инструментальное*.

Системное ПО – нужно для обеспечения работы компьютера и вычислительных сетей (создание операционной среды функционирования других программ; обеспечение надежной и эффективной работы самого компьютера и вычислительной сети; диагностика и профилактика аппаратуры компьютера и вычислительных сетей; выполнение вспомогательных технологических процессов: копирование, архивация, восстановление файлов и др.).

Инструментальное ПО – используется в ходе разработки, корректировки или развития других программ: редакторы, компиляторы, отладчики, вспомогательные системные программы, графические пакеты и др.

Прикладное ПО – программное обеспечение, состоящее из отдельных прикладных программ, библиотек программ, пакетов прикладных программ, интегрированных систем, предназначенное для решения различных прикладных задач пользователей.

Некоторые классы прикладных программ и систем:

- текстовые процессоры,
- электронные таблицы,
- информационно-поисковые системы,
- веб-обозреватели,
- базы данных,
- системы деловой графики,
- оболочки,
- интегрированные системы типа Microsoft Office.

По мере развития ПО происходил переход:

Отдельные программы → Библиотеки программ → Пакеты прикладных программ → Интегрированные системы

Отдельная программа (может быть модульной)

- пишется на некотором языке программирования
- обращение к ней осуществляется с помощью штатных средств системы программирования
- настройка на конкретную задачу осуществляется с помощью параметров

Библиотека программ – совокупность программ для решения *типовых задач* некоторой прикладной области.

Свойства программ библиотеки:

общее программное обеспечение,
единообразие способов внутренней организации и использования.

Пакет прикладных программ – комплекс взаимосвязанных прикладных программ (библиотека) и системных средств для решения некоторого класса *характерных задач* прикладной области.

Интегрированная система – комплекс программ (элементами которого могут быть отдельные программы, библиотеки, пакеты прикладных программ, системные средства) для решения *комплексных (сложных, составных, разнородных) задач*.

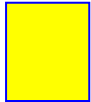
Основные составляющие классической интегрированной системы (период 1990 - 2000 гг.):

- база данных,
- электронная таблица,
- подсистема деловой графики,
- текстовый процессор.

Система деловой графики – система, поддерживающая работу с различными видами графиков и диаграмм: гистограммы, круговые и секторные диаграммы и т.д.

Оболочка – программа, создаваемая для упрощения работы со сложными программными системами. Оболочки преобразуют неудобный командный пользовательский интерфейс в дружелюбный графический интерфейс или интерфейс типа меню. Обычно оболочка реализуется в виде отдельной программы.

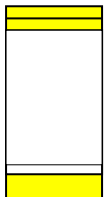
На схеме показан – выделен **желтым** цветом – примерный объем текста (фрагменты программы и/или текста на языке управления заданиями, описание входных данных), который приходится писать пользователю при создании отдельной программы и при использовании библиотек программ и пакетов прикладных программ.



При написании отдельной программы пользователь полностью описывает алгоритм решения задачи и входные данные



При использовании библиотеки пользователь описывает взаимодействие ее программ (созданных автором библиотеки заранее) и входные данные



При использовании пакета прикладных программ пользователю достаточно задать только входные данные и выбрать метод решения задачи

Проиллюстрируем эту схему на примере Задания № 1 из книги: Трифонов Н.П., Пильщиков В.Н. Задания практикума на ЭВМ (1 курс). Учебное пособие, 2-е исправленное издание. – М.: МГУ, 2001.

ПОСТАНОВКА ЗАДАЧИ

С заданной точностью eps вычислить площадь плоской фигуры, ограниченной тремя кривыми, уравнения которых $y=y_1(x)$, $y=y_2(x)$ и $y=y_3(x)$ определяются вариантом задания.

При решении задачи необходимо:

- с некоторой точностью $eps1$ вычислить абсциссы точек пересечения кривых, используя предусмотренный вариант задания метод приближенного решения уравнения $F(x)=0$; отрезки, где программа будет искать точки пересечения и где применим используемый метод, определить вручную;
- представить площадь заданной фигуры как алгебраическую сумму определенных интегралов и вычислить эти интегралы с некоторой точностью $eps2$ по квадратурной формуле, предусмотренной вариантом задания.

Величины $eps1$ и $eps2$ подобрать вручную так, чтобы гарантировалось вычисление площади фигуры с точностью eps .

Варианты задания различаются:

1. Методами приближенного решения уравнений (метод деления отрезка пополам, метод хорд, метод касательных, комбинированный метод);
2. Методами приближенного вычисления определенных интегралов решения – квадратурными формулами (формула прямоугольников, формула трапеций, формула Симпсона).
3. Набором функций (y_1 , y_2 , y_3).

Студенты первого курса должны написать программу (на языке Паскаль) для решения одного из вариантов задачи. В описании задания сформулированы требования к программе:

1. В программе предусмотреть печать как площади заданной фигуры, так и абсцисс точек пересечения кривых.

2. Описать в программе процедуру $root(f,g,a,b,eps,x)$, вычисляющую с точностью eps корень x уравнения $f(x)=g(x)$ на отрезке $[a,b]$. (Если используется метод касательных или комбинированный метод, то у $root$ должны быть еще параметры $f1$ и $g1$ - производные функций f и g .)

3. Описать в программе функцию $integral(f,a,b,eps)$, вычисляющую с точностью eps величину определенного интеграла от функции $f(x)$ на отрезке $[a,b]$.

4. Процедуру $root$ и функцию $integral$ следует предварительно протестировать.

В разделе «Методические указания» даются пояснения и указания, которые следует учесть при разработке алгоритма и его реализации:

Для корректного применения предложенных методов приближенного решения уравнения $F(x)=0$ (где $F(x)=f(x)-g(x)$) необходимо найти отрезок $[a,b]$, на котором уравнение имеет ровно один корень. Достаточное условие для этого таково: на концах отрезка функция $F(x)$ имеет разные знаки и на всем отрезке производная функции не меняет знак. Кроме того, для методов хорд и касательных, а также комбинированного метода обязательно требуется, чтобы на данном отрезке первая и вторая производные функции не меняли свой знак (не обращались в ноль).

В *методе деления отрезка пополам* определяется средняя точка c отрезка $[a,b]$ и из двух отрезков $[a,c]$ и $[c,b]$ выбирается тот, на концах которого функция $F(x)$ имеет разные знаки. К выбранному отрезку применяется та же процедура. Процесс деления отрезков прекращается, когда длина очередного отрезка станет меньше требуемой точности eps ; за корень уравнения можно принять любую точку этого отрезка. В остальных трех методах приходится учитывать свойства производных заданных функций.

Квадратурная формула для приближенного вычисления интеграла I от функции $F(x)$ на отрезке $[a,b]$ по формуле прямоугольников имеют следующий вид (n - число разбиений отрезка $[a,b]$):

$$I \cong I_n = h(F_0 + F_1 + \dots + F_{n-1}), \quad \text{где } F_i = F(a + (i+0.5)h), \quad h = (b-a)/n$$

Для обеспечения требуемой точности eps при приближенном вычислении интеграла I по квадратурной формуле нужно подобрать соответствующее число n разбиений отрезка интегрирования. Для достижения требуемой точности обычно используется следующий метод: берется некоторое начальное число разбиений n_0 (например, 10 или 20) и последовательно вычисляются значения I_n при n , равном $2n_0$, $4n_0$, $8n_0$ и т.д. Известно *правило Рунге*

$$|I - I_n| \cong p |I_n - I_{2n}|$$

(для формулы прямоугольников $p=1/3$). Согласно этому правилу, когда на очередном шаге величина $p|I_n - I_{2n}|$ окажется меньше eps , в качестве приближенного значения для I можно взять I_n или, что лучше, I_{2n} .

Если бы при выполнении задания можно было воспользоваться некоторой *библиотекой*, в состав которой входят программы:

- **приближенного решения уравнений** (метод деления отрезка пополам, метод хорд, метод касательных, комбинированный метод);
- **приближенного вычисления определенных интегралов** (формула прямоугольников, формула трапеций, формула Симпсона),

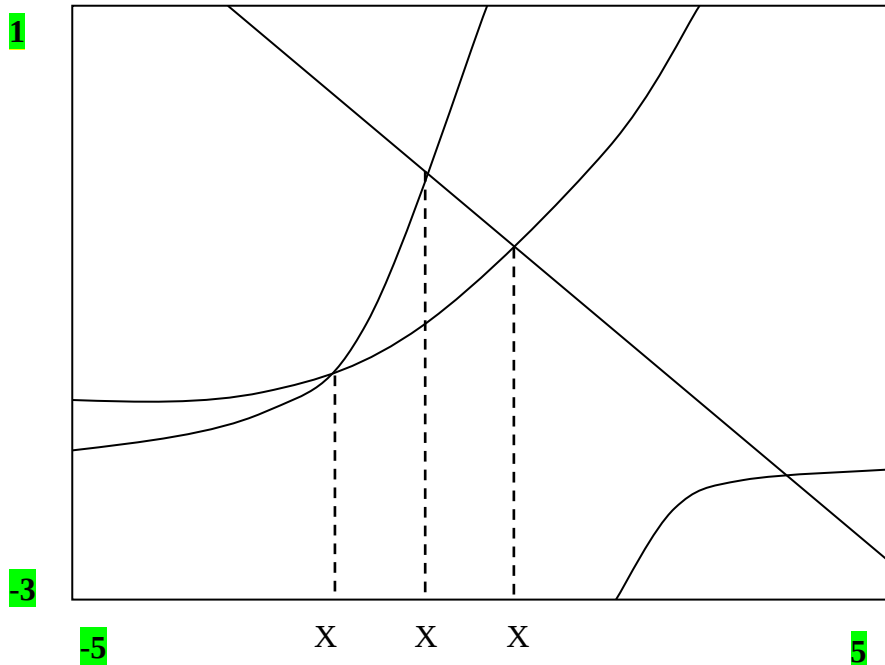
объем работы существенно уменьшился бы. Автору программы нужно было бы выбрать (для указанного варианта задания) версии процедуры $root$ и функции $integral$ и написать относительно небольшую по объему управляющую программу.

Использование подходящего пакета прикладных программ позволяет еще более существенно сократить объем работы.

Вот как выглядит диалог (некоторые мелкие детали опущены) пользователя, решающего задачу на вычисление площади криволинейного треугольника, при применении пакета прикладных программ **MathCAD** – системы, которая используется для автоматизации инженерных расчетов. Текст, набираемый пользователем на клавиатуре, выделен полужирным шрифтом и зеленым фоном. Этот текст и ответы системы (координатная сетка, графики функций, результаты вычислений) появляются на экране.

В системе используется метафора листа бумаги (бесконечного снизу и справа), на котором отображается существенная для решения задачи информация.

```
x := -5, -4.95 .. 5
y1(x) := exp(x) + 2
y2(x) := -2 x + 8
y3(x) := -5 / x
```



```
x := -3
x13 := root(y1(x) - y3(x))
```

x13 = -2.391

```
x := -1
x23 := root(y2(x) - y3(x))
```

x23 = -0.550

```
x := 1
x12 := root(y1(x) - y2(x))
```

x12 = 1.252

```
sq := ∫x13x23 y3(x) dx + ∫x23x12 y2(x) dx - ∫x13x12 y1(x) dx
```

sq = 9.807

Пять поколений языков общения с машиной. Языки 4GL и 5GL

В компьютерных науках выделяют пять поколений языков общения с машиной (Generation Languages – GL):

1GL – машинные языки

2GL – автокоды (языки ассемблера)

3GL – языки программирования высокого уровня

4GL – простейшие языки запросов информационных систем, языки запросов, ЯОД, ЯМД (баз данных), языки генераторов отчетов, языки поддержки принятия решений (статистический / прогнозный анализ), языки генераторов приложений (RPG – Report Program Generator)

5GL – должны обеспечивать автоматическое построение прикладных программ на основании данных, которые вводятся в компьютер в виде максимально удобном и привычном для пользователя, не знакомого с программированием.

Считается, что переход к каждому новому поколению повышает производительность работы с компьютером примерно на порядок (в 10 раз).

Качественное отличие языков 4GL и 5GL от языков предыдущих поколений – заключается в том, что эти языки служат *не для описания алгоритмов*, а *для спецификации решаемых пользователем задач* (описывается, *что* нужно получить, но не описывается, *как* это получить).

Функциональные требования к 4GL:

интегрированное множество функций,
язык очень высокого уровня,
совместимость с языками 3GL,
дружественный интерфейс,
средства анализа и моделирования данных,
высокая эффективность при сильной нагрузке,
поддержка стандартных СУБД

Интеллектуализация ЭВМ

Внедрение программного обеспечения, позволяющего конечному пользователю решать на ЭВМ *любые* задачи, возникающие в сфере его профессиональной деятельности без каких бы то ни было посредников (в том числе, программистов).

Понятие об автоматическом синтезе программ

Автоматический синтез программ – автоматическое/автоматизированное построение программы по описанию ее назначения или действия, называемому обычно *спецификацией программы*.

Спецификация программы vs. спецификация задачи.

Три базовых подхода к автоматическому синтезу программ:

- дедуктивный;
- индуктивный (в том числе, синтез по примерам работы программы);
- трансформационный (в том числе, синтез по прецедентам).

"Судьба" автоматического синтеза программ.

В начале исследований в этой области обычно предполагалось, что на вход поступает сформированная пользователем спецификация необходимой ему программы, а по окончании работы синтезатора выдается эта программа. Такая схема предполагает неизменность исходной спецификации (то есть невозможность дополнения и исправления спецификации в процессе синтеза), а следовательно, ее точность и полноту.

Написать подобную спецификацию программы зачастую сложнее, чем написать эту программу.

Мы вспомним об этой проблеме на следующей лекции в связи с системой ЛИСС.