## Курсовая работа

## на тему: Разработка программного обеспечения решения нелинейных уравнений.ВВЕДЕНИЕ

Язык программирования Pascal, разработанный Н.Виртом еще в 1970 г. в настоящее время получил весьма широкое распостранение. Этому способствовали его простота, удобное представление всех структурных конструкций, пригодность для использования как в учебных целях, так и для написания серьезных программ, высокая степень модульности, упрощающая процесс прграммирования, результативность автоматической проверки составленных программ на их корректность через развитый институт типов данных, высокое быстродействие трансляторов, эффективность объектных модулей. Рассматриваемый язык программирования хорошо приспособлен для решения широкого круга задач. Он содержит мощные средства структурирования данных.

# Персональные компьютеры являются наиболее широко используемым видом компьютеров, их мощность постоянно увеличивается, а область применения расширяется. Персональные компьютеры могут объединяться в сети, что позволяет десяткам и сотням пользователей легко обмениваться информацией и одновременно получать доступ к общим базам данных. Средства электронной почты позволяют пользователям компьютеров с помощью обычной телефонной сети посылать текстовые и факсимильные сообщения в другие города и странны и получать информацию из крупных банков данных.

В настоящее время почти вся современная научная деятельность человека связана с программированием. Но одной из наиболее неотрывно связанных с вычислительной техникой наук является дискретная математика, нашедшая применение в современной вычислительной технике и кибернетике: в теоретическом программировании, при проектировании ЭВМ на ЭВМ и сетей ЭВМ, баз данных, систем логического управления.

1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

* 1. Особенности взаимодействия разноязыковых модулей
     1. Проблемы комплексирования

Рассмотрим комплексирование модулей, составленных на различных языках программирования. Для САПР наиболее употребимы универсальные языки ассемблера, ФОРТРАН, ПЛ/1. Особенности операций с разноязыковыми модулями можно свести к двум группам:

1) особенности установки программной среды и вызова модуля;

2) различия в реализации типов и структур данных, проявляющиеся при обмене информацией между модулями.

1.1.2. Установка среды

При передаче управления из модуля, составленного на одном языке, в модуль, составленный на другом языке, требуется устанавливать программную среду, представляющую собой совокупность программ обработки прерываний и аварийных завершений, установки регистров, содержащих адреса областей, использующихся на протяжении выполнения задачи и др. Поэтому при организации вызова модулей, составленных на различных языках, необходимо всякий раз устанавливать среду вызываемого модуля. Рассмотрим процесс установления среды в языках ассемблера, ФОРТРАН, ПЛ/1.

В алгоритмическом языке ФОРТРАН среда устанавливается модулем IBCOM, находящимся в библиотеке компилятора. Помимо установки среды этот модуль выполняет операции ввода-вывода и др. Для программ, не содержащих операций ввода-вывода и прерываний, среда может не создаваться. Вызов модуля, составленного на языке ФОРТРАН, и установка среды этого языка представляют собой различные, самостоятельные действия.

В алгоритмическом языке ПЛ/1 установка среды обеспечивается совокупностью модулей библиотечных, сгенерированных компилятором ПЛ/1, а также построенных пользователем. Место расположения программной среды определяется вызовом модуля. Таким образом, установка среды в языке ПЛ/1 и вызов самого модуля представляют собой неразрывное целое и считаются одной операцией.

Для модулей, написанных на языке ассемблера, компилятор среды не создает. Чтобы обеспечить нормальное функционирование программы для обработки программных прерываний и аварийных ситуаций, пользователь должен сам включить в свою программу соответствующие средства, например макрокоманды SPIE, STAE, ABEND и др. Механизм задания среды определяется также пользователем, который располагается в любом месте исходного модуля макрокоманды задания среды.

Способ обращения к модулю определяется языковыми средствами организации связи по управлению, которое реализуется через активизацию модуля и возврат управления.

Средства передачи управления алгоритмических языков проиллюстрированы в таблице. /1/

|  |  |  |  |
| --- | --- | --- | --- |
| Алгоритмиче-ские языки | Характеристика средств передачи управления | | |
| Средство | Этап соединения | Тип загрузки |
| ПЛ/1 | CALL | Компиляция  Редактирование | Предварительно  По запросу |
| Ассемблера | ATACH  LINK  FETCH  XCTL  CALL | Выполнение  Компиляция  Редактирование | Предварительно  По запросу |
| ФОРТРАН | CALL  Вызов  Функции | Компиляция  Редактирование | Предварительно  По запросу |

1.1.3. Согласование типов

Различные языки программирования обладают разными наборами типов данных. Возможные отношения между типами данных приведены ниже.

К эквивалентным типам данных относятся типы данных, для которых внутреннее представление, сгенерированное компиляторами, идентично.

К косвенно-эквивалентным типам данных относятся типы данных, для которых нет эквивалентного описания, но с помощью имеющихся языковых средств их можно совместить.

К неэквивалентным типам данных относят данные, которые невозможно свести друг к другу с помощью имеющихся языковых средств.

Рассматриваемые алгоритмические языки различаются не только типами, а также заданиями длины отдельных элементов данных. Длина элеметов данных может указываться явно или неявно (по умолчанию). Кроме того, необходимо отметить следующие организационные особенности каждого языка:

1) обратное расположение массивов (в языке ФОРТРАН массив располагается в памяти по столбцам, а в языке ПЛ/1 - по строкам);

2) наличие информационного вектора для переменных в языке ПЛ/1 [поскольку в языке ПЛ/1 память под данные может выделяться динамически, компилятор генерирует информационный вектор, который содержит сведения о переменных, строках и массивах (адрес, длина, количество элементов и пр.)] и имеет различный вид для различных структур данных;

3) выравнивание полей, которое определяется тем, что компилятор располагает данные в соответствии с описаниями, выравнивая их, если необходимо, на границы слов и полуслов. /1/

1.1.4. Обмен данными

Наиболее распространен способ обмена данных с помощью операторов CALL и LINK. При этом осуществляется формирование списка передаваемых данных и списка их адресов. Адрес списка адресов передается вызываемой программе через регистр 1.

Другим распространенным способом обмена данными является использование общих областей - статически распределеных участков памяти, к которым может обращаться любой модуль независимо от того, на каком языке он написан. Память под общую область отводит редактор связей во время создания загрузочной программы из совокупности общих областей отдельных модулей. Каждый из рассматриваемых языков имеет средства для описания общих областей: в языке ФОРТРАН - оператор COMMON, в языке ПЛ/1 - оператор STATIC EXTERNAL, в языке ассемблера - оператор COM. /1/

1.1.5. Информационный адаптер

С учетом вешеизложенного для осуществления универсального информационного интерфейса необходимо:

1) произвести контроль наличия исходных данных для каждого отдельного модуля;

2) задать недостающие исходные данные;

3) проверить соответствие типов, структур и последовательности данных аналогичным характеристикам данных, принятым в вызываемом модуле;

4) преобразовать данные в случае несоответствия типов;

5) обеспечить передачу данных вызываемому модулю в соответствии с типом обмена;

6) организовать среду, определяемую языком программирования модуля;

7) проверить резулбтаты;

8) выполнить обратное преобразоавание данных в вид, принятый для хранения промежуточных результатов;

9) сохранить результаты работы модуля для дальнейшего использования;

Реализация описанных функций выполняется программой-адаптером. Информационное обеспечение адаптера (рисунок 1.1) включает в себя таблицы описателей модулей, наборов данных, область обмена.

Монитор

МФ

МВв

ТОМ

ТОВх

МВыв

ОО

ПУС

ТОНД

Програмный

адаптер

МФ - модули

формирования; МВв - модули ввода; ТОНД - таблица описателей наборов данных; ТОМ - таблица описателей модулей; ПУС - программа установки среды; ТОВх - таблица описателей входов; МВыв - модули вывода; ОО - области обмена;

Рисунок 1.1 - Информационное обеспечение программного адаптера

Таблица описателей модулей содержит: имя модуля; идентификатор языка программирования; признак типа обмена, принятый в модуле (параметры, общие области, наборы данных); количество параметров; имя описателей входов модуля; имя области обмена; имя описателя набора данных.

Таблица описателей входов модуля содержит: имя параметра локальное; имя параметра глобальное; характеристику параметра (входной, выходной, модифицируемый); вид структуры (переменная строка, массив арифметический, массив строк, структура, массив структур и т.д.); размерность (для массива); длину (для строк); основание системы счисления (для переменной или элемента массива); форму представления; точность.

Таблица описателей наборов данных содержит: имя набора; имя DD-предложения; тип организации; метод доступа; формат и длина блока; атрибуты записи.

Для обеспечения преемственности данных следует предусмотреть область памяти, куда заносились бы те результаты работы каждого модуля, которые принимают участие в дальнейших расчетах. Этой цели служит область обмена - память, выделенная для промежуточных результатов, передаваемых от модуля к модулю. В нее заносятся исходные данные для первого модуля, результаты вычислений, недостающие данные для промежуточных модулей.

Для организации области обмена необходимо:

1) осуществить анализ данных, которые будут циркулировать в среде программного комплекса в составе САПР с целью определения одинаковых по смыслу переменных;

2) всем одинаковым по смыслу переменным присвоить одно и то же имя, отражающее семантику этих переменных (глобальные переменные);

3) каждой переменной, встречающейся только один раз, присвоить уникальное имя, отражающее семантику величины (локальные переменные);

4) каждой глобальной и локальной переменной поставить в соответствие тип и структуру, являющиеся наиболее информативными;

Указанная совокупность переменных составляет область обмена, в которой хранится текущее значение каждой из переменных в наиболее информативном виде.

В крупных САПР, программы которых оперируют с большим числом входных, промежуточных и результирующих переменных, области обмена удобно организовывать в виде некоторого банка данных. Это позволяет возложить часть функций, выполняемых адаптером, на СУБД, что в конечном итоге сокращает время на разработку информационного и программного обеспечения САПР.

Таким образом, адаптер выполняет всю совокупность операций по организации информационного взаимодействия между программными модулями. В случае разноязыковых модулей адаптер практически берет на себя выполнение соответствующих функций операционной системы. Достаточно сложной является также задача построения области обмена, поскольку ее решение связано со структурированием всех переменных, участвующих в информационном обмене. В крупных САПР, программные модули которых оперируют с большим числом входных, промежуточных и результирующих переменных, функции адаптера по организации и взаимодействию с обменными областями целесообразно переложить на типовые СУБД.

Банки данных в настоящее время находят все более широкое применение для организации межмодульного интерфейса. Их использование наиболее эффективно, когда совокупность модулей программного обеспечения зафиксирована и не подлежит изменениям в дальнейшем. В этом случае необходимо составить логическую схему для всей области обмена, в которой были бы указаны наименования переменных, их взаимосвязи, тип представления. Обращение из программных модулей для получения значений необходимых переменных должно выполняться с помощью операторов взаимодействия с СУБД. Применение банков данных для целей организации информационного обмена сокращает сроки разработки информационного и программного обеспечения САПР. /1/

2. АЛГОРИТМИЧЕСКАЯ ЧАСТЬ

2.1. Математическое решение задачи

Пусть даны два уравнения с двумя неизвестными

F1(x,у)=0, (1)

F2(x,у)=0

действительные корни которых требуется найти с заданной степенью точности.

Мы предположим, что система (1) допускает лишь изолирован­ные корни. Число этих корней и их грубо приближенные значения можно установить, построив кривые F1(x,у)=0; F2(x,у)=0 и определив координаты их точек пересечения.

Пусть *х=x0; у=y0-*приближенные значения корней системы (1), полученные графически или каким-нибудь другим способом (на­пример, грубой прикидкой).

Дадим итерационный процесс, позволяющий при известных усло­виях уточнить данные приближенные значения корней. Для этого представим систему (1) в виде

x=ϕ1(x,y),

y=ϕ2(x,y)

и построим последовательные приближения по следующим форму­лам:

x1=ϕ1(x0,y0); y1=ϕ2(x0,y0);

x2=ϕ1(x1,y1); y1=ϕ2(x1,y1); (3)

xn+1=ϕ1(xn,yn); yn+1=ϕ2(xn,yn)

Если итерационный процесс (3) сходится, т. е. существуют пре­делы

ξ=lim xn и η=lim yn,

n→∞ n→∞

то, предполагая функции ϕ1(x,y) и ϕ2(x,y) непрерывными и пере­ходя к пределу в равенстве (3) общего вида, получим:

lim xn+1=lim ϕ1(xn,yn)

n→∞ n→∞

lim xn+1=lim ϕ2(xn,yn)

n→∞ n→∞

Отсюда ξ=ϕ1(ξ,η); η=ϕ2(ξ,η)

т. е. предельные значения ξ и η являются корнями системы (2), а следовательно, и системы (1). Поэтому, взяв достаточно большое число итераций (3), мы получим числа xn и yn*,* которые будут отличаться от точных корней x=ξ и y=η системы (1) сколь угодно ма­ло. Поставленная задача, таким образом, окажется решенной. Если итерационный процесс (3) расхо­дится, то им пользоваться нельзя.

**Теорема**. Пусть в некоторой замкнутой окрестности R {a**≤**x≤A; b≤y≤B}(рис.) имеется одна и только одна пара корней x=ξ и y=η системы (2). Если:1) функции ϕ1(x,y) и ϕ2(x,y) определены и непрерывно дифференцируемы в R; 2) начальные при­ближения x0, y0 и все последующие приближения xn, yn (n=1,2...) принадлежат R; 3) в R выполнены неравенства

⏐∂ϕ1/∂x⏐+⏐∂ϕ2/∂x ⏐≤q1<1

⏐∂ϕ1/∂x⏐+⏐∂ϕ2/∂x ⏐≤q2<1

то процесс последовательных приближений (3) сходится к корням x=ξ и y=η системы (2), т.е.

ξ=lim xn и η=lim yn,

n→∞ n→∞

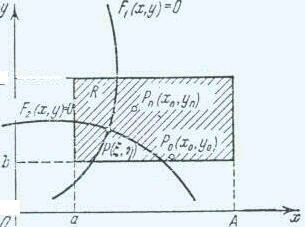


Рисунок 2.1-Графики уравнений в замкнутой окрестности.

Структурная схема решения задачи представлена на рисунке 2.2.

Начало

Вывод наглядного изображения функции на дисплей

Обнуление массивов и используемых переменных

Установка

X0 и Y0

i:=i+1

Вычисление

Xi и Yi

Установка

X0 и Y0

Конец

Достигнута точность

Нет Да

Рисунок 2.2-Структурная схема решения задачи.

3. РАЗРАБОТКА СТРУКТУРЫ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

3.1. Построение структуры программного обеспечения

При программировании использовался язык программирования Turbo Pascal. Текст программы приведен в приложении A. На рисунке 3.1 представлена структура программы, решающей поставленную задачу. Программа имеет монолитно-модульную структуру.

Нахождение решения системы уравнений

Установка начальных параметров

Метод

итераций

Вывод полученного решения

Интерфейс

Рисунок 3.1-Монолитно-модульная структура программы

Опишем некоторые основные стандартные процедуры и функции, используемые для формирования собственных процедур.

Для установки (инициализации) графического режима служит функция initgraph ( ).

InitGraph(gd,gm,'');

(gd**-**graphic driver, gm-graphic mode);

Функция initgraph( )считывает в память соответствующий драйвер, устанавливает видеорежим, и определяет маршрут к директории, в которой находится драйвер egavga.bgi.

При работе с цветом активно использовалась функция SetColor (int color), выставляющая цвет надписи и фоновый цвет. В программе с помощью этой функций выводились следующие цвета:

1. синий - SetColor(LightBlue);

2) красный **-** SetColor(LightRed);

3) белый **-** SetColor (White);

4) зеленый - SetColor(LightGreen);

Функция outtextxy (int x, int y, textstring text )позволяет осуществить вывод текстовой строки textstring,начиная с точки с координатами x,y.

Для очистки экрана от информации использовалась функция ClearScreen(ClrScr).Функция CloseGraphв конце процедурывыводит систему из графического режима.

В программе используется следующие типы констант:

**-** integer **-** целочисленные значения; **-** real **-** действительные значения;

**-** text **-** тип текстового файла; Модули pcx,pcx1**-**графические,используются для установления параметров экрана. В частности, в следующих строках устанавливается режим работы VGAhi.

gd:=VGA; gm:=vgahi;

У этого режима работы следующие характеристики:

-разрешение: 640 на 480

-число цветов: 16;

Также вместе с запускным файлом должны находится следующие файлы:

- egavga.bgi **-** EGA/VGA Video Driver

3.2. Описание диалога с пользователем

При запуске программы на экране монитора рисуется график требуемой в задаче функции “Лемнискаты Бернули”. После первого нажатия клавиши Enter, программа показывает первоначальные приближения функции и окончательные значения x и y. После второго нажатия клавиши Enter программа завершает работу.

ЗАКЛЮЧЕНИЕ

В процессе написания данной курсовой работы, были приобретены навыки работы с пакетом программ MS Office 2000, в частности, с текстовым редактором Word 2000, что, наверняка, облегчит дальнейшее общение с компьютером. Были получены новые знания в таких науках, как "Технология программирования" и "Дискретная математика".

Сама программа была написана на языке программирования TurboPascal, являющемся сейчас одним из самых распространенных в мире, наряду с Си и Си++.

При подготовке данной работы использовалось техническое обеспечение со следующими характеристиками:

-компьютер AMD K5-100/16/6.4 Gb/SVGA 2 Mb/FDD 3.5/14' LG

-принтер Epson Stylus Color 600

Список использованных источников

1. Норенков И.П. Системы автоматизированного пректирования: Учебное пособие для ВТУЗов: в 9 кн/Кн. 3: Федорук В.Г. Черненький В.М. Информационное и пограмное обеспечение. - М.: Высшая школа, 1986.-159 с.

2. Технология разработки программных средств: Методические указания / Э.И.Воробьёв, О.Ю.Макаров, А.В.Антиликаторов; Воронеж. гос.тех. ун-т.- Воронеж, 1997.- 24 с.

3. Курсовое проектирование. Организация, порядок проведения, оформление расчётно-пояснительной записки и графической части: Стандарт предприятия / Г.Д. Дель; Воронеж. гос. тех. ун-т. - Воронеж, 1998.- 48 с.