**Тема «Основы компьютерной алгебры»**

**Лабораторная работа №1**

**Работу выполнила: Белорукова Елизавета Игоревна**

**Студентка 2 курса ИВТ, 1 подгруппа**

**Компьютерная обработка информации: модели, методы, средства.**

**Обработка информации** — получение одних «информационных объектов» (структур данных) из других путём выполнения некоторых алгоритмов.

**Исполнитель алгоритма** — абстрактная или реальная система, способная выполнить действия, предписываемые алгоритмом. Для механизации и автоматизации процесса обработки информации и вычислений, выполняемых в соответствии с заданным алгоритмом, используют различные типы вычислительных машин: механические, электрические, электронные (ЭВМ), гидравлические, пневматические, оптические и комбинированные.

В современной информатике основным исполнителем алгоритмов является ЭВМ, называемая также **компьютером** (от англ. computer — вычислитель).

**ЭВМ** — электронное устройство, предназначенное для автоматизации процесса алгоритмической обработки информации и вычислений.

В зависимости от формы представления обрабатываемой информации вычислительные машины делятся на три больших класса:

* **Цифровые вычислительные машины** (ЦВМ), обрабатывающие информацию, представленную в цифровой форме;
* **Аналоговые вычислительные машины (**АВМ), обрабатывающие информацию, представ-

ленную в виде непрерывно меняющихся значений какой-либо физической величины (электрического напряжения, тока и т.д.);

* **Гибридные вычислительные машины** (ГВМ), содержащие как аналоговые, так и цифровые вычислительные устройства.

Современный компьютер (ЭВМ) как реальная система обработки данных имеет ряд особенностей:

1. ЭВМ располагает конечным множеством команд, лежащих в основе реализации и выполнения каждого алгоритма;
2. ЭВМ функционирует дискретно (потактно) под управлением программы, хранящейся в оперативной памяти;
3. ЭВМ имеет широкий набор команд, что позволяет эффективно представлять разнообразные алгоритмы решаемых задач;
4. Каждая ЭВМ является потенциально универсальной. Потенциальность объясняется тем, что ни одна ЭВМ не может считаться универсальной в смысле вычислимости произвольной, частично рекурсивной функции, т.е. для неё существует класс нерешаемых задач при условии неизменности её ресурсов (в первую очередь памяти).

При выполнении задач обработки информации на компьютере выделяют **пакетный и интер-**

**активный** (запросный, диалоговый) режимы взаимодействия пользователя с ЭВМ.

**Пакетный режим** первоначально использовался для снижения непроизводительных затрат машинного времени путём объединения однотипных заданий. Его суть заключается в следующем. Задания группируются в пакеты, каждый со своим отдельным компилятором. Компилятор загружается один раз, а затем осуществляется последовательная трансляция всех заданий пакета. По окончании компиляции пакета все успешно транслированные в двоичный код задания последовательно загружаются и обрабатываются. Такой режим был основным в эпоху централизованного использования ЭВМ (централизованной обработки), когда различные классы задач решались с использованием одних и тех же вычислительных ресурсов, сосредоточенных в одном месте (информационно вычислительном центре). При этом организация вычислительного процесса строилась главным образом без доступа пользователя к ЭВМ. Его функции ограничивались лишь подготовкой исходных данных по комплексу информационно-взаимосвязанных задач и передачей их в центр обработки, где формировался пакет заданий для ЭВМ.

В настоящее время под ***пакетным режимом*** также понимается *процесс компьютерной обработки заданий без возможности взаимодействия с пользователем*. При этом, как правило, задания вводятся пользователями с терминалов и обрабатываются не сразу, а помещаются сначала в очередь задач, а затем поступают на обработку по мере высвобождения ресурсов. Такой режим реализуется во многих системах коллективного доступа.

**Интерактивный режим** предусматривает непосредственное взаимодействие пользователя с информационно-вычислительной системой и может носить характер запроса (как правило, регламентированного) или диалога с ЭВМ. **Запросный режим** позволяет дифференцированно, в строго установленном порядке предоставлять пользователям время для общения с ЭВМ. **Диалоговый режим** открывает пользователю возможность непосредственно взаимодействовать с вычислительной системой в допустимом для него темпе работы, реализуя повторяющийся цикл выдачи задания, получения и анализа ответа. При этом ЭВМ сама может инициировать диалог, сообщая пользователю последовательность шагов (предоставление меню) для получения искомого результата. Взаимодействие пользователя и ЭВМ осуществляется путём передачи сообщений и управляющих сигналов между пользователем и ЭВМ.

Входные сообщения генерируются оператором с помощью средств ввода: клавиатуры, манипуляторов типа мышь и т.п., выходные — компьютером в виде текстов, звуковых сигналов, изображений и представляются пользователю на экране монитора или других устройствах вывода информации.

Основными типами сообщений, генерируемыми пользователем, являются: запрос информации, запрос помощи, запрос операции или функции, ввод или изменение информации и т.д. В ответ со стороны компьютера он получает: подсказки или справки, информационные сообщения, не требующие ответа, приказы, требующие действий, сообщения об ошибках, нуждающиеся в ответных действиях и т.д.

**Данный режим является основным** на современном этапе развития компьютерных систем обработки информации, характерной чертой которого является широкое внедрение практически во все сферы деятельности человека персональных компьютеров (ПК) — однопользовательских микроЭВМ, удовлетворяющих требованиям общедоступности и универсальности применения. В настоящее время пользователь, обладая знаниями основ информатики и вычислительной техники, сам разрабатывает алгоритм решения задачи, вводит данные, получает результаты, оценивает их качество. У него имеются реальные возможности решать задачи с альтернативными вариантами, анализировать и выбирать с помощью системы в конкретных условиях наиболее приемлемый вариант.

**Основные этапы решения задач с помощью компьютера:**

***1.Постановка задачи:***

сбор информации о задаче;

формулировка условия задачи;

определение конечных целей решения задачи;

определение формы выдачи результатов;

описание данных (их типов, диапазонов величин, структуры и т.п.).

***2.Анализ и исследование задачи, модели:***

анализ существующих аналогов;

анализ технических и программных средств;

разработка математической модели;

разработка структур данных.

***3.Разработка алгоритма:***

выбор метода проектирования алгоритма;

выбор формы записи алгоритма (блок-схема, псевдокод и др.);

выбор тестов и метода тестирования;

проектирование алгоритма.

***4.Программирование:***

выбор языка программирования;

уточнение способов организации данных;

запись алгоритма на выбранном языке программирования.

***5.Тестирование и отладка:***

синтаксическая отладка;

отладка семантики и логической структуры;

тестовые расчёты и анализ результатов тестирования;

совершенствование программы.

6.Анализ результатов решения задачи и уточнение в случае необходимости математической модели с повторным выполнением этапов 2—5.

***7.Сопровождение программы:***

доработка программы для решения конкретных задач;

составление документации к решенной задаче, математической модели, алгоритму, программе, по их использованию.

**Структуры данных в компьютерной алгебре.**

**Структурой данных** называется совокупность множеств {M1, M2, ... MN} и совокупность отношений {P1, P2, ... PR}, определённых над элементами этих множеств: S={M1,M2,...MN ;P1,P2,...PR}

Бинарное отношение, задающее массив – **орграф.**

Структура данных линейна, если орграф не содержит циклов и может быть изображен в виде одной̆ линии.

**Память вычислительной̆ (алгоритмической̆) машины имеет линейную структуру.**

Обработка любого типа информации (имеющего структуру произвольной̆ сложности) должна моделироваться на схеме массива – линейной̆ структуре.

**Линейная структура памяти – вектор памяти.**

Отношение «иметь имя» переопределяется с помощью отношения «иметь адрес». Адрес произвольного элемента массива

вычисляется по формуле: ai = a0 + i \* b (a0 – база, адрес 1-го элемента массива; i – номер адресуемого элемента; b – число ячеек, занимаемых одним элементом массива).

**Экземпляром структуры** данных называется совокупность IS = { Mai, V, P, val },

где Mai – множество элементов ai;

V – множество значений;

P – множество отношений следования; val – отношение «иметь значение».

**Схемой структуры** данных называется совокупность SS = { Mai, P },

где Mai – множество элементов ai;

P – множество отношений следования.

Одной̆ SS может соответствовать множество ES. Алгоритм реализуется над схемой, а конкретные вычисления (преобразования) по алгоритму производятся над экземплярами.

Операции над структурами данных :

* Создание и уничтожение структуры данных;
* Поиск элемента данных в структуре;
* Обновление структуры данных: вставка нового и удаление старого элемента;
* Обход структуры данных с выполнением определённых, наперёд заданных действий.

**Рекурсивным** называется **список**, элементами которого могут являться списками.

**Рекурсивные списки** способны представлять данные любого уровня структурной̆ сложности.

Элементами рекурсивного списка могут быть не списки.

**Операции над списками**

Пусть L = (A1, A2, A3, ... Ai, ... An) – список. Тогда определены следующие операции:

**Базовые:**

* Создание нулевого списка: L = ( )
* Получение 1-го элемента (головы) списка: A1 для списка L
* Получение остатка списка (переход по ссылке к следующему элементу):

(A2, A3, ... Ai, ... An) для списка L

* Конкатенация (слияние) двух списков L1 и L2: L = (L1 , L2)

**Дополнительные:**

* Получение следующего элемента Ai+1 , если известен предыдущий Ai
* Вставка нового элемента B после элемента Ai, т.е. получение из списка L =

(A1, A2, A3, ... Ai, Ai+1, ... An) нового списка M = (A1, A2, A3, ... Ai, B, Ai+1, ... An)

* Удаление элемента, следующего за элементом Ai, т.е. получение из списка L = (A1, A2, A3, ... Ai, Ai+1, ... An) нового списка

M = (A1, A2, A3, ... Ai, Ai+2, ... An)

**Базовые типы данных:**

* Числа
* Математические выражения

Числа произвольной точности представляются в виде массивов, последовательностей и списков.

**Выводы :**

**1)** Вся память машины символьных вычислений состоит из ячеек.

**2)** Каждая ячейка входит в состав определённого списка.

**3)** Начальная конфигурация – один список свободного места, объединяющий̆ все ячейки.

**4)** Каждая следующая конфигурация – это результат операции над данными (при этом требуется перераспределение ячеек – изменение указателей̆):

* создание нового списка для вновь поступивших данных;
* увеличение длины списка (за счёт первой̆ ячейки списка свободного места);
* уменьшение длины списка (освобождение некоторых, ранее занятых ячеек; освобождаемые ячейки присоединяются к голове списка свободного места);
* уничтожение списка (освобождение всех ячеек списка, которые таким же образом пополняют список свободного места).

**Системы компьютерной алгебры: достижения и перспективы**

1. Расширение состава встроенных и программируемых типов математических объектов;
2. Интеграция СКА с другими компьютерными системами;
3. Унификация и объектная ориентация интерфейса пользователя;
4. Программирование символьных вычислений произвольной̆ сложности;
5. Ускорение работы СКА.

* Принадлежность математического объекта СКА к встроенным должна определяться не случайной̆ практической̆ необходимостью (СКА ранних поколений), а ролью в иерархической̆ системе математических абстракций (СКА Axiom).
* Способность к созданию расширений (объектных, структурных, функциональных и т.п.) СКА должна поддерживаться интерфейсом (желательно с помощью объектно- ориентированного, специализированного языка программирования) (СКА Maple, СКА Mathematica).
* Интеграция ядра и расширений СКА должна быть прозрачной̆ для любого пользователя СКА (цель пользователя – решение прикладной̆ задачи, а не организация взаимодействия вычислительных модулей̆).

**Связь с программами числовой̆ обработки:**

Связь типа «СКА(СВМ)» или типа СВМ(СКА): вставка «машинных» кодов программ на процедурных языках в тело программ аналитических вычислений или наоборот (язык С и СКА Mathematica).

Связь типа «СКА+СВМ»: обмен результатами вычислений с помощью файлов.

Связь нецелесообразна: разработка специализированной̆ системы смешанных

(численно-аналитических) вычислений.

**Генерация текста программ вычислений:**

Поддержка не одного (СКА Reduce – язык Fortran), а нескольких (СКА Maple, СКА

Mathematica – языки Fortran и C) целевых языков программирования.

Сложно-структурированные математические выражения наглядно создаются в СКА, а затем без ошибок (!) транслируются в строковую форму операторов присваивания.

**Связь с текстовыми процессорами:**

Поддержка в СКА общепринятых форматов нетекстовых объектов – формул, графиков, рисунков – обеспечивает либо полную, либо частичную вёрстку научных документов (TEX-формат).

Для унификации пользовательский̆ интерфейс СКА должен иметь те же функциональные возможности, что и интерфейсы других сред программирования и проектирования

(настройка параметров, редактирование объектов, отладка проектов и т.п.).

Для объектной̆ ориентации необходима реализация специальных классов объектов, представляющих алгебраические и другие абстрактные математические категории

(тождества, многообразия, исчисления и т.п.).

Для образовательных и рекламных целей̆ требуется наличие инструментальных средств создания интерактивных документов (анимационная графика, панели управления и т.п.).

Увеличение количества встроенных в СКА библиотек шаблонов пользовательских приложений для различных предметных областей̆ (СКА Maple).

Использование в качестве языка реализации СКА - функционально расширяемого языка программирования (LISP), обеспечивающее не только неограниченный̆ рост сложности вновь создаваемых приложений, но и совершенствование базовых объектов и алгоритмов аналитических вычислений (СКА Mathematica).

Постоянное совершенствование способов символьного представления математических объектов и алгоритмов выполнения аналитических преобразований.

Применение технологии JIT-компиляции машинных кодов для реализации наиболее трудоёмких операций компьютерной̆ алгебры (решение дифференциальных уравнений, статистическое моделирование и т.п.).