

ЛЕКЦИЯ 1

Структуры и алгоритмы
обработки данных



1.1 Типы данных, абстрактные типы и структуры данных

Типы данных включают:

- **натуральные и целые числа;**
- **вещественные (действительные) числа (в виде приближенных десятичных дробей);**
- **литеры;**
- **строки и др.**

Защита типов может быть более или менее жесткой:

- - язык Pascal сохраняет весьма строгую защиту типов. Pascal-компилятор в большинстве случаев расценивает смешение в одном выражении данных разных типов или применение к типу данных несвойственных ему операций как фатальную ошибку.
- - язык C является языком со слабой защитой типов и в случае смешения типов данных C-компиляторы выдают предупреждения.

- **Абстрактный тип данных (АТД)** – это математическая модель и набор операторов, определенных в рамках данной модели.
- Базовым строительным блоком структуры данных является **ячейка**, которая предназначена для хранения значения определенного базового или составного типа данных.

1. 2 Классификация структур данных

- **Физическая структура** отражает способ физического представления данных в памяти машины и называется еще структурой хранения, внутренней структурой или структурой памяти.
- **Абстрактная или логическая структура** – это структура данных без учета ее представления в машинной памяти.

Различают:

- простые (базовые, примитивные) структуры (типы) данных;
- интегрированные (структурированные, композитные, сложные).

В зависимости от отсутствия или наличия явно заданных связей между элементами данных, следует различать:

- **- несвязные структуры** (векторы, массивы, строки, стеки, очереди)
- **- связанные структуры** (связные списки).

**По признаку изменчивости
различают структуры:**

- **статические,**
- **полустатические,**
- **динамические.**

**В зависимости от
упорядоченности элементов
структуры данных делят на:**

- - линейные,
- нелинейные.

По признаку взаимного расположения элементов в памяти линейные структуры можно разделить:

- - на структуры с последовательным распределением элементов в памяти (векторы, строки, массивы, стеки, очереди)
- - структуры с произвольным связным распределением элементов в памяти (односвязные, двусвязные списки).

Информация по каждому типу однозначно определяет:

- структуру хранения данных указанного типа, т.е. выделение памяти и представление данных в ней, с одной стороны, и интерпретирование двоичного представления, с другой;
- множество допустимых значений, которые может иметь тот или иной объект описываемого типа;
- множество допустимых операций, которые применимы к объекту описываемого типа.

Статические структуры

Статические структуры относятся к разряду непримитивных структур, которые являются множеством примитивных базовых структур, память для них выделяется автоматически.

Статические структуры в языках программирования связаны со структурированными типами.

Структурированные типы позволяют строить структуры данных сколь угодно большой сложности. К таким типам относятся: массивы, записи и множества (этот тип реализован не во всех языках).

Классификация структур данных

Рис.1

Структуры данных				
Простые базовые структуры	Статические структуры	Полустатические структуры	Динамические структуры	Файловые структуры
Числовые, Символьные, Логические, Перечисление, Интервал, Указатели	Векторы, Массивы, Множества, Записи, Таблицы	Стеки, Очереди, Деки, Строки	Линейные связанные списки, Разветвленные связанные списки, Деревья, Графы	Последовательного, Прямого, Комбинированного доступа, Организованные разделами, индексированные

1.3 Виды памяти, указатели и работа с ними в языке Pascal (Delphi)

Различают два способа распределения памяти:

- **статическое** – во время трансляции программы, что эффективно, поскольку в ходе выполнения программы на управление памятью не расходуется ни время, ни память;
- **динамическое управление памятью**, которое осуществляется во время выполнения программы, а переменные, которые создаются и уничтожаются в этом процессе, называются динамическими. Для организации динамической памяти используется тип данных, называемый **указателем** или **ссылочным типом данных**.

Указатели

Значением указателя является **адрес области памяти**, содержащей переменную заранее определенного типа. В этом случае указатели называются **типизированными**.

Нетипизированный указатель служит для представления адреса, по которому содержатся данные неизвестного типа.

Для указателей область памяти выделяется **статически**, а для переменных, на которые они указывают, – **динамически**.

Наиболее частые случаи, когда могут понадобиться указатели:

1. При необходимости представить одну и ту же область памяти, а, следовательно, одни и те же физические данные, как **данные разной логической структуры**;

2. При работе с динамическими структурами данных

При объявлении типизированного указателя определяется и тип объекта в памяти, адресуемого этим указателем. Для объявления переменных используется символ « ^ », после которого указывается тип динамической (базовой) переменной.

Type <имя_типа> = ^ <базовый тип>;

Var <имя_переменной>: <имя_типа>; или
<имя_переменной>: ^ <базовый тип>:

Например:

Type ss = ^ Integer;

Var x, yt: ss; { *Указатели на переменные целого типа * }

z : ^Real; { *Указатель на переменную вещественного типа * }

•

Выделение оперативной памяти для **динамической переменной базового типа** осуществляется с помощью процедуры $New(x)$, где x определен как *соответствующий указатель*.

Обращение к динамическим переменным выполняется по правилу: $\langle \text{имя переменной} \rangle ^ \wedge$.

Например: $x^\wedge := 15$.

Здесь в область памяти (два байта), адрес которой является значением указателя x , записывается число 15.

Процедура $Dispose(x)$ освобождает память, занятую динамической переменной. При этом значение указателя x становится неопределенным.

Основными операциями, в которых участвуют указатели:

1. присваивание;
2. получение адреса;
3. выборка.

Присваивание – это двухместная операция, оба операнда которой – указатели. Как и для других типов, операция присваивания копирует значение одного указателя в другой, в результате чего оба указателя будут содержать один и тот же адрес памяти. Типизированные указатели должны ссылаться на объекты одного и того же типа.

Операция получения адреса – одноместная, ее операнд может иметь любой тип, результатом является типизированный (в соответствии с типом операнда) указатель, содержащий адрес объекта-операнда.

Операция выборки – одноместная, ее операндом является типизированный указатель, результат – данные, выбранные из памяти по адресу, заданному операндом. Тип результата определяется типом указателя-операнда.

Схемы, иллюстрирующие работу с указателями:

