Типизация и структуризация данных

Литература

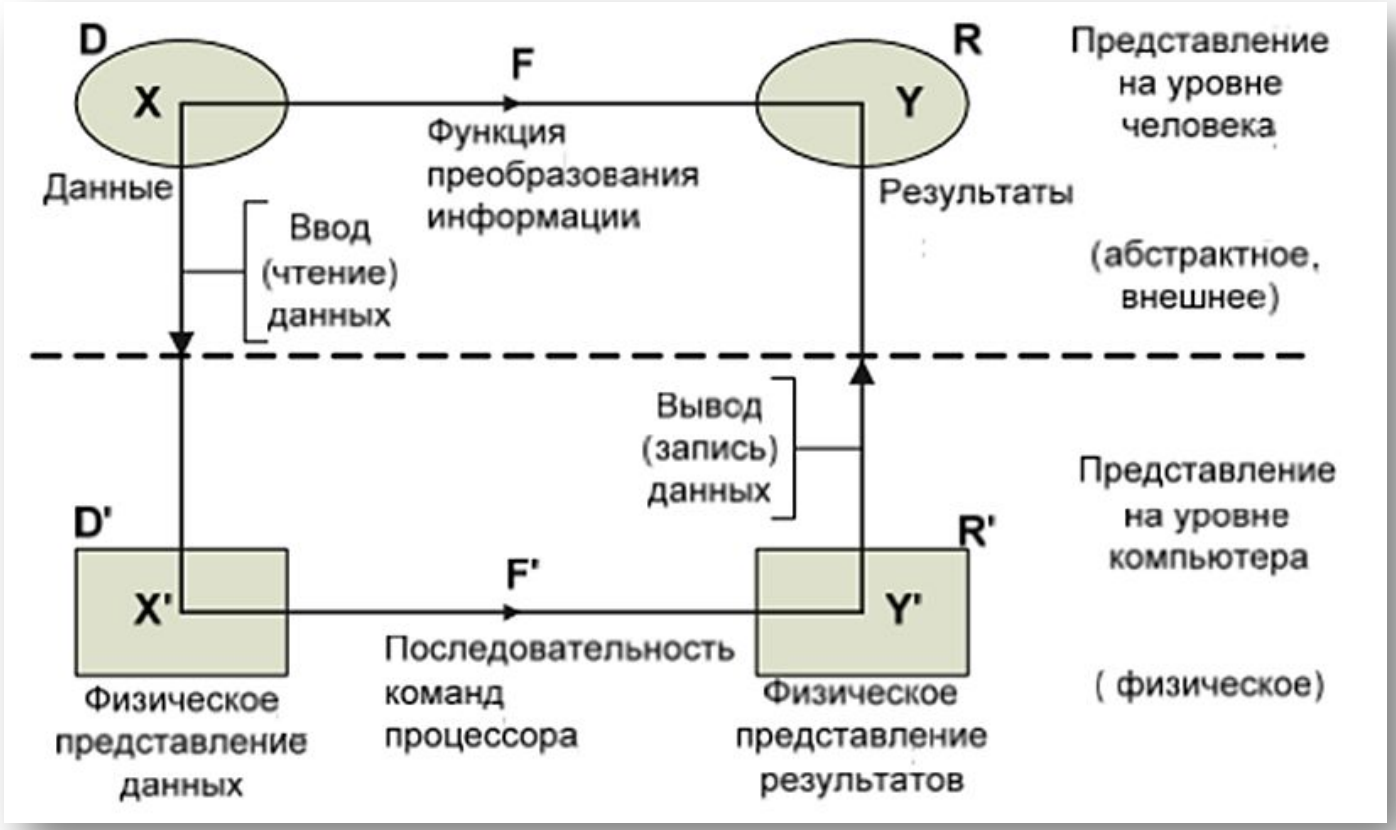
1. Вирт Н. Алгоритмы + структуры данных = программы. - М.: Мир.
2. ГОСТ 20886-85 Организация данных в системах обработки данных. Термины и определения

Организация данных

- **Данные** – это представление фактов и идей в формализованном виде, пригодном для передачи и переработке в некоем процессе
- **Информация** - это смысл, который придается данным при их представлении

- **Организация данных** – представление данных и управление данными в соответствии с определенными соглашениями.

Модель обработки информации



- Обработка информации – это практическая реализация некоторой функции F, которая отображает множество данных D во множество возможных результатов R.
  - F – произвольная функция, которую надо «вычислить», например, перевод текста с русского на английский, нахождение максимума, расчет траектории ракеты, построение оптимального плана и т.д.

Организация данных

- **Представление данных** (Data representation) – характеристика, выражающая
  - правила кодирования элементов
  - и образования конструкций данных на конкретном уровне рассмотрения в вычислительной системе
- **Управление данными** (Data management) – совокупность функций обеспечения
  - требуемого представления данных,
  - накопления и хранения,
  - обновления и удаления,
  - поиска по заданному критерию и выдачи данных

Организация данных

- **Представление данных** (Data representation) – характеристика, выражающая
  - правила кодирования элементов
  - и образования конструкций данных на конкретном уровне рассмотрения в вычислительной системе
- **Управление данными** (Data management) – совокупность функций обеспечения
  - требуемого представления данных,
  - накопления и хранения,
  - обновления и удаления,
  - поиска по заданному критерию и выдачи данных

При постановке задачи необходимо выбрать некоторое абстрактное представление предмета рассмотрения, т.е. определить **множество данных**, отражающих реальную ситуацию (модель предметной области).

Уровни организации данных

- Логическая организация данных: проектный уровень
  - отражает взгляд пользователя на данные
  - применяются формальные методы описания динамически изменяющихся структур
- Представление данных: уровень языка реализации
  - описание данных на языке программирования
- Физическая организация данных
  - учитывается размещение и связь данных в среде хранения

Понятие о типизации языка

Тип объекта

- С машинной точки зрения

Форма представления его значений в памяти.

Определяется способ доступа к объекту и его части.

- С точки зрения разработчика

множество значений и набор операций, выполняемых над этими значениями и обладающих некоторыми свойствами

Контроль типов

- Основная функция типов
  - обеспечение более полной и легкой проверки правильности программ.
- Проверка заключается
  - в определении типов выражений
  - и их согласованности с типами, которые требуются по правилам языка.

Такая проверка называется ***контролем типов***.

Правила типизации

Программа называется *типово-правильной*, если она удовлетворяет правилам типизации языка:

- приписывание типов переменным и константам,
- определение типов выражений по типам их частей,
- согласование типов частей языковых конструкций.

Язык программирования является ***типизированным***, если для него определены правила типизации.

Статическая типизация

- переменная, параметр подпрограммы, возвращаемое значение функции связывается с типом в **момент объявления** и тип не может быть изменён позже
  - Ада, Си++, Паскаль

Динамическая типизация

- переменная связывается с типом **в момент присваивания значения**, а не в момент объявления переменной
  - Python, Ruby, PHP, Perl, JavaScript

Уровни типизации

- **Слабо типизированный** (нестрогая типизация) – если информация и типе используется только для обеспечения корректности программы на машинном уровне (ПЛ/1, Алгол-68, Си и С++)
  - разрешается выполнение некорректных операций
  - повышает гибкость языка, но уменьшает понятность и надежность программ.

- **Сильно типизированный** (строгая типизация) – если осуществляется полный контроль типов (язык Ада, C#)
  - повышает надежность и ясность программ

Преимущества типизации

- Модель предметной области лучше структурирована, существует иерархия сортов элементов
- Манипулирование элементами более целенаправленно, разнородные элементы обрабатываются различным образом, однородные – единообразно
- В случае строгой типизации несоответствия типов фиксируются до выполнения программы, гарантируя отсутствие смысловых ошибок и безопасность кода

Тип данных

Определяет

- Формат представления в памяти компьютера
- Множество допустимых значений, которые может принимать принадлежащая к выбранному типу переменная или константа
- Множество допустимых операций, применимых к этому типу.

Простые и структурные типы данных

- ***Простые***
  - Целочисленные
  - Вещественные
  - Логический тип
  - Символьный тип
- ***Структурированные***
  - Массив
  - Структура
  - Перечисление
  - Класс

Типы данных C++

Название	**Обозначение**	**Диапазон значений**
Байт	char	от -128 до +127
без знака	unsigned char	от 0 до 255
Короткое целое число	short	от -32768 до +32767
Короткое целое число без знака	unsigned short	от 0 до 65535
Целое число	int	от – 2147483648 до + 2147483647
Целое число без знака	unsigned int (или просто unsigned)	от 0 до 4294967295
Длинное целое число	long	от – 2147483648 до + 2147483647
Длинное целое число без знака	unsigned long	от 0 до 4294967295
Вещественное число одинарной точности	float	от ±3.4e-38 до ±3.4e+38 (7 значащих цифр)
Вещественное число двойной точности	double	от ±1.7e-308 до ±1.7e+308 (15 значащих цифр)
Вещественное число увеличенной точности	long double	от ±1.2e-4932 до ±1.2e+4932
Логическое значение	bool	значения true(истина) или false (ложь)

Тип C#	Размер в байтах	Тип .NET	Описание
Базовый тип			
object		Object	Может хранить все что угодно, т.к. является всеобщим предком
Логический тип			
bool	1	Bolean	true или false
Целые типы			
sbyte	1	SByte	Целое со знаком (от -128 до 127)
byte	1	Byte	Целое без знака (от 0 до 255)
short	2	Int16	Целое со знака (от -32768 до 32767)
ushort	2	UInt16	Целое без знака (от 0 до 65535)
int	4	Int32	Целое со знаком (от -2147483648 до 2147483647)
uint	4	UInt	Целое число без знака (от 0 до 4 294 967 295)
long	8	Int64	Целое со знаком (от -9223372036854775808 до 9223372036854775807)
ulong	8	UInt64	Целое без знака (от 0 до 0ffffffffffffff)
Вещественные типы			
float	4	Single	Число с плавающей точкой двойной точности. Содержит значения приблизительно от $\pm 1.5*10^{-45}$ до $\pm 3.4*10^{38}$ с 7 значащими цифрами
double	8	Double	Число с плавающей точкой двойной точности. Содержит значения приблизительно от $\pm 5.0*10^{-324}$ до $\pm 1.7*10^{308}$ с 15-16 значащими цифрами
Символьный тип			
char	2	Char	Символы Unicode
Строковый тип			
string		String	Строка из Unicode-символов
Финансовый тип			
decimal	12	Decimal	Число до 28 знаков с фиксированным положением десятичной точки. Обычно используется в финансовых расчетах.

Структурированные типы

Необходимость в структурных типах данных

- Для разработки программ методом сверху вниз необходимо иметь возможность описывать данные на различных уровнях.
- Данные должны быть структурированы, чтобы их можно было эффективно выбирать.

Общее понятие структуры данных

- **Абстрактный тип данных** (АТД):

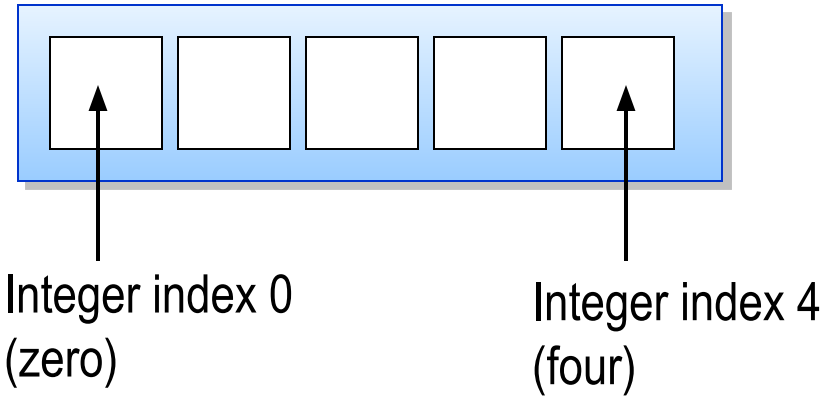
математическая модель и операции, определенные в рамках этой модели.

Для представления АТД используются **структуры данных**, которые представляют собой набор переменных, возможно различных типов, объединенных определенным образом.

- **Абстрактные структуры данных** предназначены для удобного хранения и доступа к информации
  - предоставляют удобный интерфейс для типичных операций с хранимыми объектами, скрывая детали реализации от пользователя

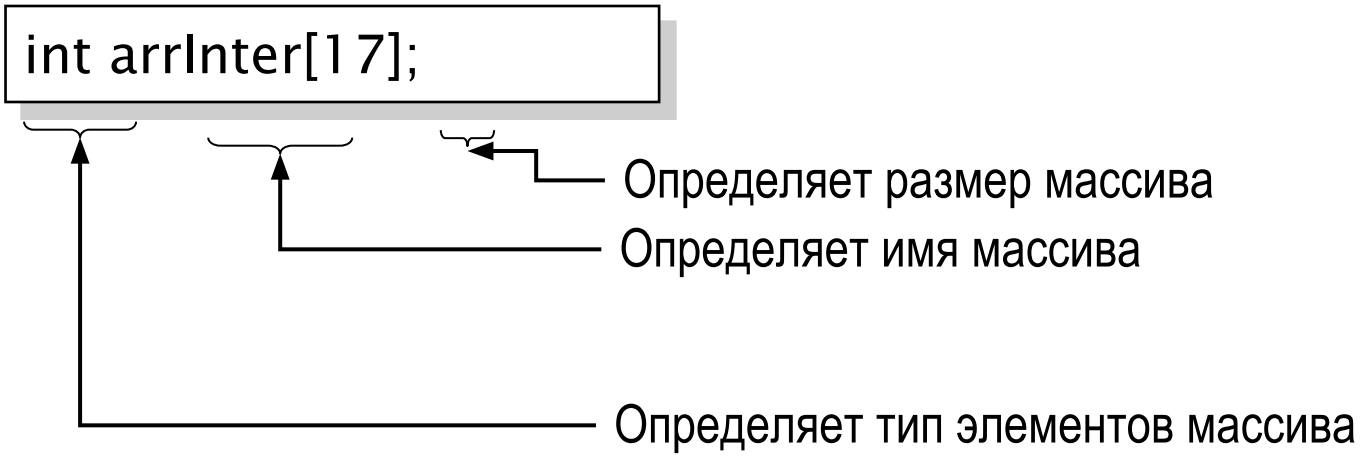
Массив

- Массив – это последовательный набор элементов
  - Все элементы массива одного типа
  - Структуры могут хранить элементы разных типов
  - Доступ к конкретным элементам массива происходит через использование индекса



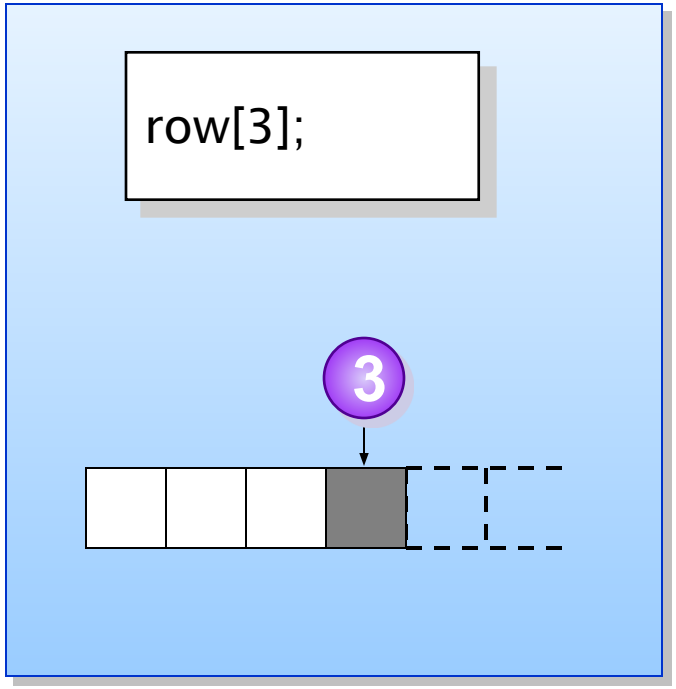
Форма записи массива в C++

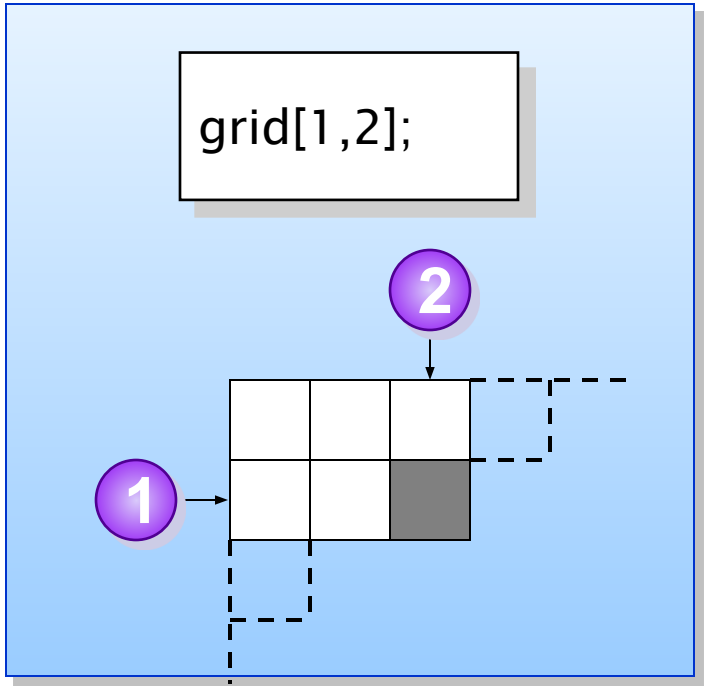
- При объявлении массива необходимо определить:
  - Тип элементов массива
  - Размер массива
  - Имя массива



Организация доступа к элементам массива

- Определяйте индекс для каждой из размерностей
  - Индекс первого элемента равен нулю





Перечисления

- Перечисляемый тип представляет собой тип значений, содержащий конечное число именованных констант
- Синтаксис определения перечисления

```
enum <имя> [ : базовый тип]
{список-перечисления констант(через запятую)};
```

- Создание перечисления

```
enum DayTime { morning, day, evening, night };
```

- Использование перечисления

```
DayTime current;
if (current != night)
    // выполнить работу
```

Структуры

- Создание структуры

```
public struct Employee
{
   string firstName;
   int age;
}
```

- Использование структуры

```
Employee companyEmployee;
companyEmployee.firstName = "Joe";
companyEmployee.age = 23;
```

Структуры могут хранить элементы разных типов

Динамические структуры данных

- Особенности:
  - отсутствие физической смежности элементов структуры в памяти
  - непостоянство и непредсказуемость размера (числа элементов) структуры в процессе ее обработки
- Элемент динамической структуры состоит из двух полей
  - информационного поля или поля данных
  - поле связок

Связное представление данных

Достоинства :

- размер структуры ограничивается только доступным объемом машинной памяти;
- при изменении логической последовательности элементов структуры требуется не перемещение данных в памяти, а только коррекция указателей;
- большая гибкость структуры.

Недостатки:

- на поля связок расходуется дополнительная память;
- доступ к элементам связной структуры может быть менее эффективным по времени.

Реализация структур данных

- В языках программирования имеется возможность явно запрашивать и использовать области **динамической памяти**.
- C++
  - **Указатель** содержит адрес поля в динамической памяти, хранящего величину определенного типа.
    - Сам указатель располагается в статической памяти
- C#
  - Коллекция – группа объектов.
  - Коллекции упрощают реализацию многих задач программирования, предлагая уже готовые решения для построения структур данных

Коллекции общего назначения

Реализуют структуры данных:

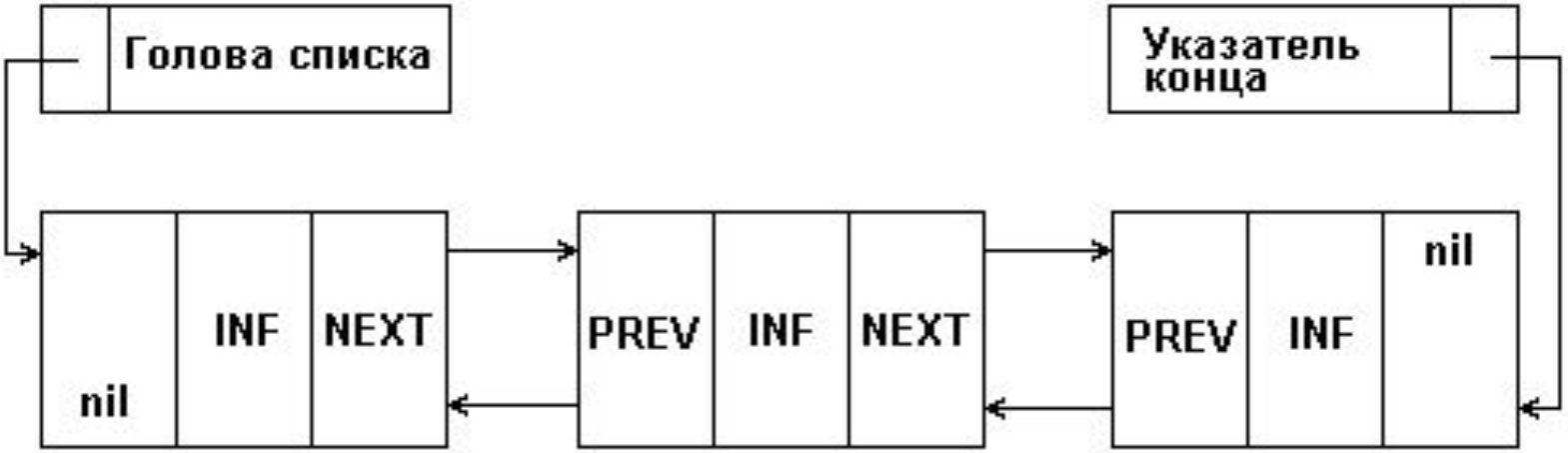
- стеки,
- очереди,
- динамические массивы,
- словари (хеш-таблицы, предназначенные для хранения пар ключ/значение),
- отсортированный список для хранения пар ключ/значение

Простейшие структуры данных

- **Список -** упорядоченное множество, состоящее из переменного числа элементов, к которым применимы операции включения, исключения.
- **Линейный список** - список, отражающий отношения соседства между элементами

Представление односвязного списка в памяти

Представление двусвязного списка в памяти



Стек

- **Стек** - такой последовательный список с переменной длиной, включение и исключение элементов из которого выполняются только с одной стороны списка, называемого вершиной стека.
- LIFO (Last – In – First - Out - "последним пришел - первым исключается").
- Основные операции над стеком
  - включение нового элемента (английское название push - заталкивать)
  - исключение элемента из стека (англ. pop - выскакивать).

Очередь FIFO

- **Очередью** FIFO (First – In – First - Out - "первым пришел - первым исключается") называется такой последовательный список с переменной длиной, в котором
  - включение элементов выполняется только с одной стороны списка (эту сторону часто называют концом или хвостом очереди),
  - а исключение - с другой стороны (называемой началом или головой очереди).
- Основные операции:
  - включение,
  - исключение,
  - определение размера, очистка,

Дек

- **Дек** - особый вид очереди.
- Дек (от англ. deq - double ended queue,т.е очередь с двумя концами) - это такой последовательный список, в котором как включение, так и исключение элементов может осуществляться с любого из двух концов списка.
- Операции над деком:
  - включение элемента справа;
  - включение элемента слева;
  - исключение элемента справа;
  - исключение элемента слева;
  - определение размера; очистка.