Абстрактные типы данных

Санкт-Петербургский
государственный университет

АТД

- АТД — некоторая математическая модель и набор операций, определённый в рамках этой модели
  - Обобщение понятия «тип»
- Состоит из типа данных и операций, выполняющих над ним преобразования
  - Внутреннее устройство типа данных невидимо для остальной программы (принцип сокрытия деталей реализации)
  - Работа с АТД — только с помощью связанных с ним функций
  - Тип данных и операции для работы с ним лежат в одном модуле, так, чтобы все изменения в АТД были локализованы и не затрагивали остальную программу (принцип инкапсуляции)
- Дальнейшее обобщение АТД — классы

Пример — стек

- stack.h / stack.c, при этом структура данных описана только в .c-файле, в .h-файле только её предварительное объявление
  - Так компилятор может гарантировать сокрытие деталей реализации
    - Всё, что не проверяется автоматически, можно считать не работающим!
  - Все функции принимают только указатель на структуру, для значения нужно знать размер
- Функции:
  - createStack()
  - deleteStack()
  - push()
  - pop()
  - isEmpty()
- Внешнему миру вообще всё равно, как стек устроен внутри
  - Может быть на массиве

Ещё пример — список

- Требуется целых два типа — сам список и позиция внутри списка
  - Что-то вроде индекса элемента массива, но может быть устроена хитрее
  - Позиция должна обеспечивать быструю работу с элементом, на который она указывает
  - Внешнему миру всё равно, как устроен список и что такое позиция
    - Может быть, список на массивах, а позиция — число, или список на указателях, а позиция — указатель на элемент списка (или даже на предыдущий элемент)
- Список может хранить разные типы элементов
  - typedef — «шаблоны для бедных»
    - ```
      typedef int Value;
      struct ListElement {
          Value value;
          ListElement* next;
      }
      ```
    - typedef же может использоваться для описания типа позиции
      ```

Инвариант

- Некоторое логическое условие, верное всё время жизни АТД
  - Не совсем, внутри функции АТД инвариант может нарушаться
- АТД отвечает за поддержание своего инварианта
  - Поскольку работа с АТД только через его функции, у внешнего мира нет способа его испортить
- Пример — размер списка
  - Можно считать за $O(n)$ каждый раз
  - Можно хранить как элемент структуры, тогда должен соблюдаться инвариант
- Ещё пример — head и tail у очереди

Пример применения АТД — сортировка слиянием

Если в списке больше одного элемента, делим его на два, вызываем mergesort, получаем два отсортированных списка, которые сливаем в один отсортированный

- $O(n * log(n))$ в среднем и худшем случае
- Устойчива
- Внешняя (подходит для больших данных, не помещающихся в память)
- https://www.hackerearth.com/practice/algorithms/sorting/merge-sort/visualize/
- Ей не надо знать внутреннего устройства списка