Списки и другие абстрактные типы данных

Лекция 8

План лекции

- Абстрактные типы данных
  - Несколько примеров
  - Определение
  - Зачем использовать АТД
- АТД список
  - Реализация на языке Си через 1-связные списки
- Другие АТД на основе списков
- Стек и примеры использования стеков
  - Перевод арифметического выражения из инфиксной в постфиксную запись
  - Вычисление значения выражения на стеке

Кто придумал абстрактные типы данных?

- Барбара Лисков р. 1939
- Стивен Жиль р. ?
- Liskov B., Zilles S. Programming with abstract data types // SIGPlan Notices, vol. 9, no. 4, 1974

- Использование метода абстракции в программировании на примере построения польской записи выражения с помощью стека

Примеры АТД 1/4

Система регулирования скорости

- Задать скорость
- Получить текущие параметры
- Восстановить предыдущее значение скорости
- Отключить систему

Кофемолка

- Включить
- Выключить
- Задать скорость
- Начать перемалывание
- Прекратить перемалывание

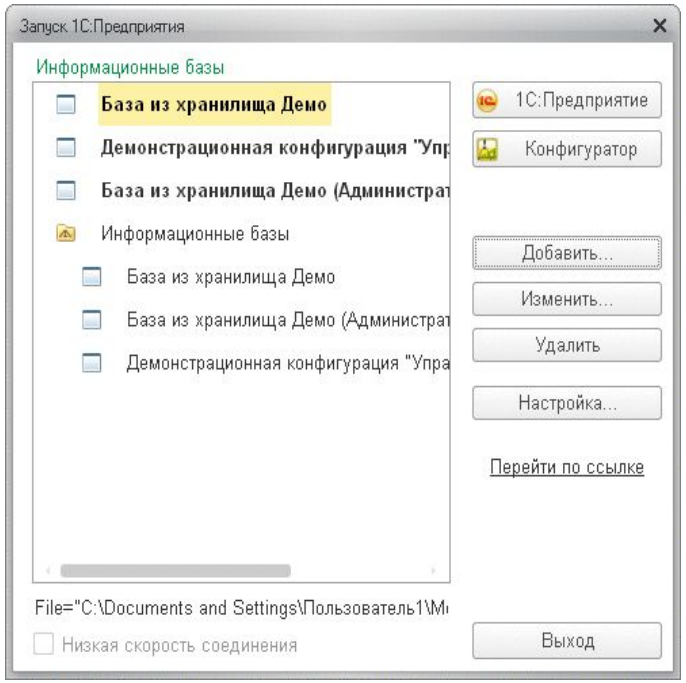
Примеры АТД 2/4

Топливный бак

- Заполнить бак
- Слить топливо
- Получить емкость топливного бака
- Получить статус топливного бака

Список

- Инициализировать список
- Вставить элемент
- Удалить элемент
- Прочитать следующий элемент



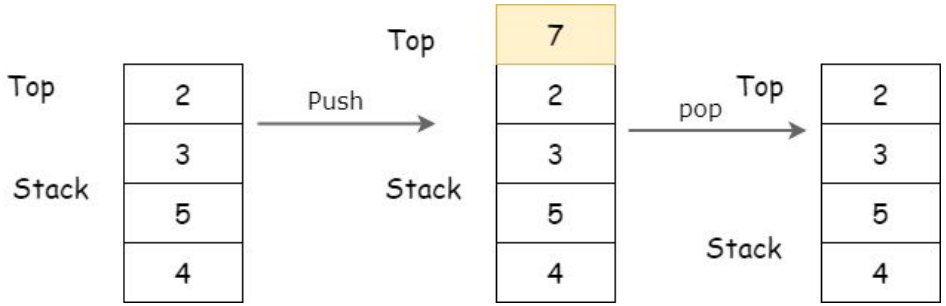
Примеры АТД 3/4

Фонарь

- Включить
- Выключить

Стек

- Инициализировать стек
- Поместить элемент
- Извлечь элемент
- Проверить наличие элементов



Примеры АТД 4/4

Указатель

- Выделить блок памяти
- Освободить блок памяти
- Изменить объем выделенной памяти

Файл

- Открыть
- Прочитать байты
- Записать байты
- Установить позицию чтения/записи
- Закрыть

Определение АТД

- Абстрактный тип данных – это множество значений и набор операций, для которых не важно представление этих значений в памяти
- АТД – это класс обычных типов данных, которые используются и ведут себя одинаково
- Реализация АТД – это один из обычных типов данных, принадлежащих классу, который задает АТД
  - Конкретный набор подпрограмм, выполняющих операции над конкретными значениями в памяти
  - Один АТД может допускать несколько принципиально разных реализаций

Контейнеры

- Контейнер – это АТД, использующийся для группировки элементов и доступа к ним

Контейнер	Другие названия	Типичная реализация
Список (list)	Последовательность, sequence	Массив, 1-связный список
Очередь (queue)	FIFO	Массив, 1-связный список
Дек (double-ended queue)	Dequeue, deque	Массив, 2-связный список
Стек (stack)	Магазин, LIFO	Массив, 1-связный список
Ассоциативный массив (associative array)	Словарь, dictionary, hash map, hash, map, хэш	Хэш-таблица
Множество (set)		Красно-черное дерево, хэш-таблица
Очередь с приоритетом (priority queue)	Пирамида, heap	Пирамида

Зачем использовать АТД?

Реализация АТД

- Сокрытия деталей реализации
- Ограничение области использования данных
- Ограничение области изменений
- Легкость оптимизации кода

Использование АТД

- Более высокая информативность интерфейса
- Работа с сущностями реального мира
  - А не с деталями реализации
- Удобочитаемость и понятность кода

АТД список

- Конечная последовательность элементов

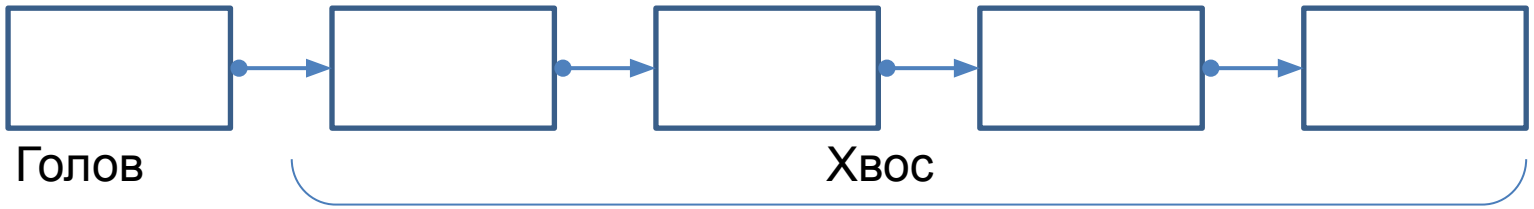
- Минимальный набор операций
  - Создать пустой список
  - Добавить элемент в начало списка
  - Удалить первый элемент списка
  - Вернуть значение первого элемента списка (головы списка)
  - Вернуть список всех элементов, кроме первого (хвост списка)
  - Проверить наличие элементов в списке

Реализации списков

- Число связей у элемента
  - 1-связные
  - 2-связные
- Топология
  - Линейная
  - С циклом

Одно- и двусвязные списки

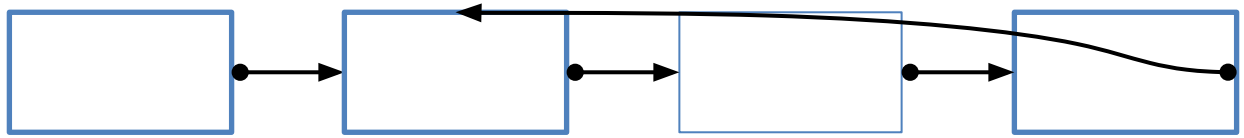
- Односвязный список – это список, каждый элемент которого имеет <= 1 соседа



- Двусвязный список – это список, каждый внутренний элемент которого имеет двух соседей

Циклические списки

- Циклический список – это список, по элементам которого можно сколь угодно долго двигаться в одну из сторон



- Как определить, является ли список циклическим, не изменяя список и не используя дополнительной памяти?
  - Почему рассматриваемый АТД список не позволяет создавать циклические списки?
  - Как сделать возможным создание циклических списков средствами АТД список?

АТД список на языке Си

```
// T -- тип элементов
// TList -- список элементов типа T

TList  Create(void);
void   Append(TList *A, T v);
void   Remove(TList *A);
T      GetHead(TList A);
TList  GetTail(TList A);
int    IsEmpty(TList A);
```

- Напишите АТД «список с закладками» (итераторами)

Пример использования АТД список

```
// Найти значение в списке
int HasValue(TList A, T v) {
    TList a = A;
    while (!IsEmpty(a)) {
        if (GetHead(a) == v) {
            return 1;
        }
        a = GetTail(a);
    }
    return 0;
}
// Перепишите с помощью for
```

Пользуемся, не зная, что внутри! ☺

☺ На экзамене так не говорить

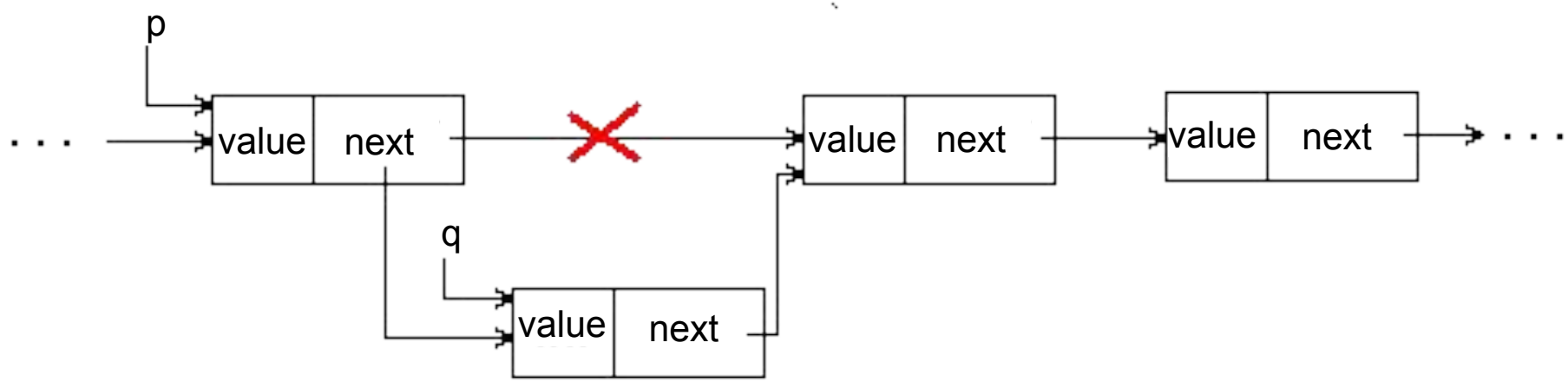
Реализация через 1-связный список

```
struct TList {
    T Value;
    struct TList* Next;
};

typedef struct TList* TList;
```

Реализация Append – добавить в начало

```
void Append(TList *A, T v) {
    TList q = malloc(sizeof *q);
    assert(q != NULL);
    q->Value = v;
    q->Next = *A;
    *A = q;
}

// Напишите функцию
// void Insert(TList *list, T value, T afterValue);
// добавляющую value после элемента, равного afterValue
```

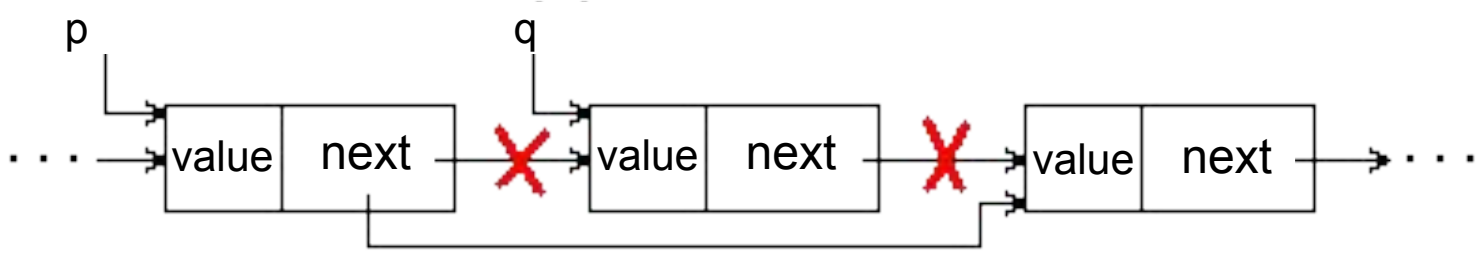
Вставка в 1-связный список в общем случае



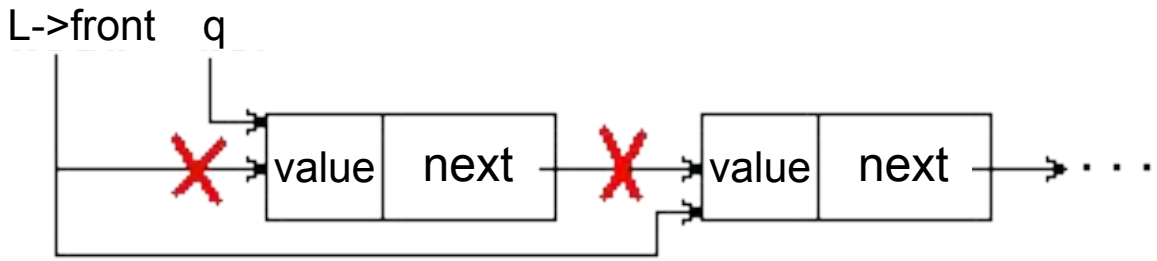
Реализация Remove – уничтожить 1й элемент

```
void Remove(TList *A) {
    assert(*A != NULL);
    TList a = (*A)->Next;
    free(*A);
    *A = a;
}

// Напишите функцию
// void Erase(TList *A, T value);
// удаляющую элемент, равный value
```

Удаление из 1-связного списка в общем случае



Из середины списка



Из начала списка

Реализация GetHead/GetTail

```
TList GetTail(TList A) {
    assert(A != NULL);
    return A->Next;
}
```

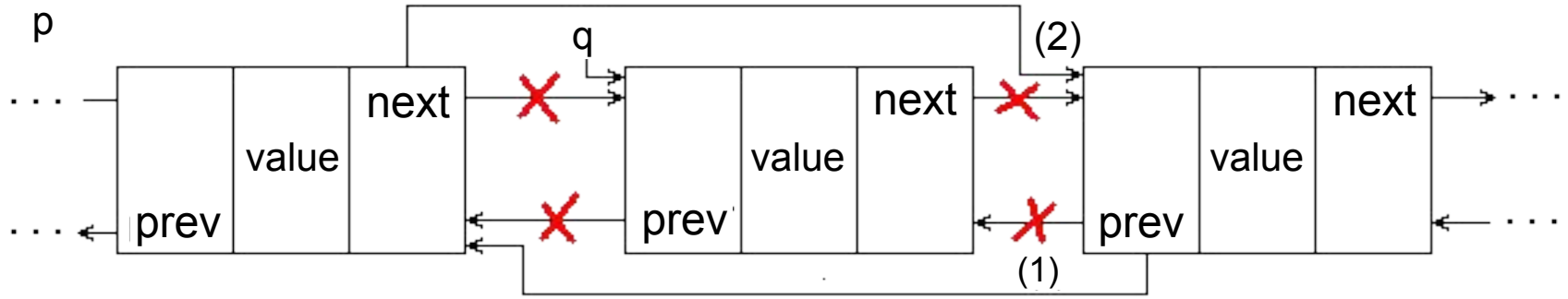
```
TList GetHead(TList A) {
    assert(A != NULL);
    return A->Value;
}
```

Реализация через 2-связный список

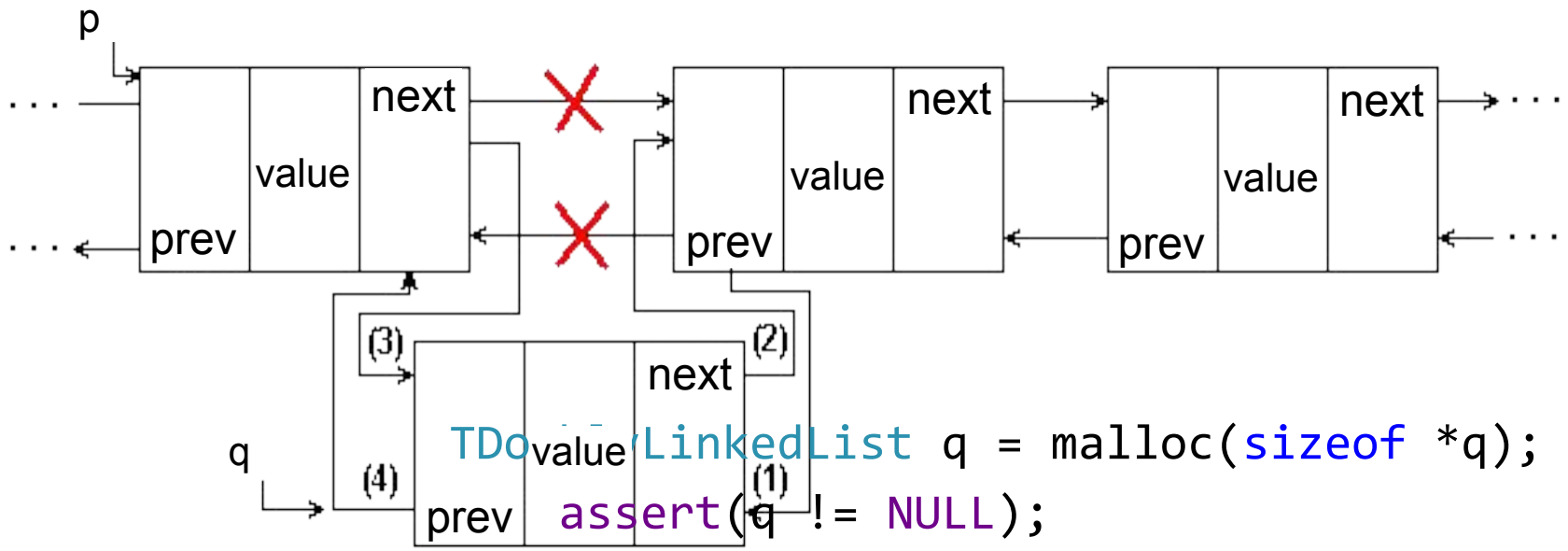
```
struct TDoublyLinkedList {
    T Value;
    struct TList* Next;
    struct TList* Previous;
};

typedef struct TDoublyLinkedList* TDoublyLinkedList;
```

Удаление из 2-связного списка

```
TDoublyLinkedList q = p->next;
    p->next->next->prev = p; // (1)
    p->next = q->next;       // (2)
    free(q);
```



Вставка в 2-связный список




```
TDoubleLinkedList q = malloc(sizeof *q);
assert(q != NULL);
p->next->prev = q; // (1)
q->next = p->next; // (2)
p->next = q;       // (3)
q->prev = p;       // (4)
```

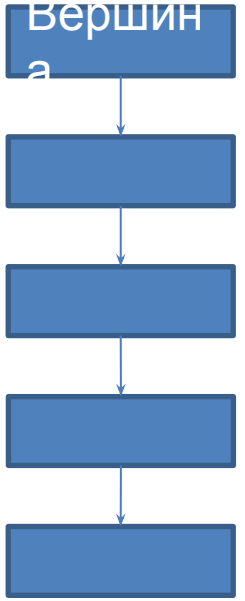
АТД на основе списков

- Стек (stack)
- Очередь (queue)
- Дек (double-ended queue)

АТД стек

- Стек -- это список, в котором добавление/удаление элементов происходит только на одном конце
- Последний добавленный в стек ячейка называется вершиной стека

- реверсивная память
- гнездовая память
- магазин
- push-down список
- LIFO (last-in-first-out)
- список йо-йо



Операции со стеком

Обозначение	Смысл
Create(S)	создать пустой стек
GetTop(S)	вернуть вершину стека
Pop(S)	вернуть вершину и удалить её
Push(S, x)	добавить новый элемент x
IsEmpty(S)	проверить наличие элементов в стеке
Destroy(S)	уничтожить стек

Обратная польская запись выражений

- Скобочная (инфиксная) запись выражения
  - a + (f – b * c / (z – x) + y) / (a * r – k)
- Обратная польская (постфиксная) запись
  - a f b c * z x – / – y + a r * k – / +

- Постфиксная запись = программа вычисления арифметического выражения
- Как из инфиксной записи получить постфиксную запись?

- Ян Лукашевич 1878-1956
  - Польский логик, родился в г. Львов
  - Использовал бесскобочную запись, нач
    «польскую

Построение обратной польской записи

- Вход: скобочная запись арифметического выражения
- Выход: обратная польская запись того же арифметического выражения

Операция	Приоритет П
( )	1
+ –	2
* /	4

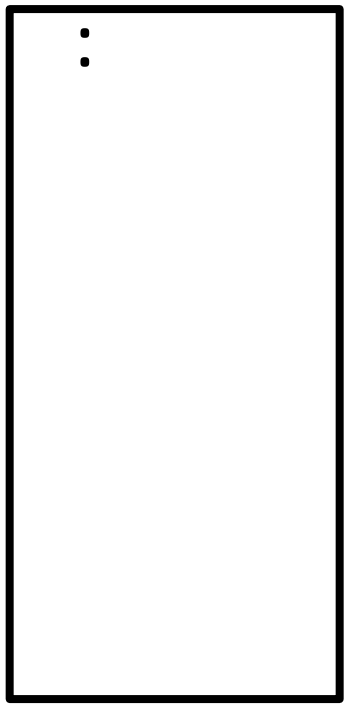
Построение обратной польской записи

```
Create(операторы), польская = «»
пока инфиксная != «» повторять
    x = первый элемент инфиксная, удалить x из инфиксная
    если x – число или переменная, то польская += x
    иначе если x = (, то Push(операторы, x)
    иначе если x = ), то
        пока GetTop(операторы) != ( повторять
            польская += Pop(операторы)
        Pop(операторы) // убрать саму (
    иначе
        пока !IsEmpty(операторы) && П(GetTop(операторы)) >= П(x) повторять
            польская += Pop(операторы)
        Push(операторы, x)
пока !IsEmpty(операторы) повторять польская += Pop(операторы)
Destroy(операторы)
```

Пример

a + (f − b * c / (z − x) + y) / (a * r − k)

Стек



X =

Выходная строка:

Вычисление по польской записи

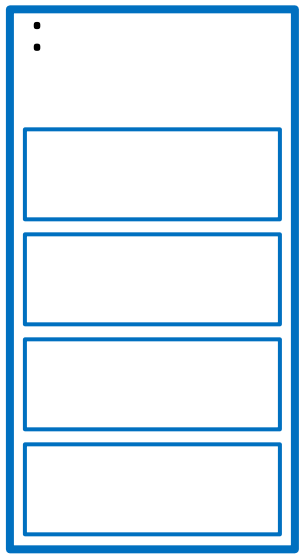
```
Create(операнды)
пока польская != «» повторять
   x = первый элемент польская, удалить x из польская
   если x – число или переменная, то
      Push(операнды, значение(x))
   если x – оператор, то
      a = Pop(операнды)
      b = Pop(операнды)
      Push(операнды, a x b)
значениеПольской = Pop(операнды)
Destroy(операнды)
```

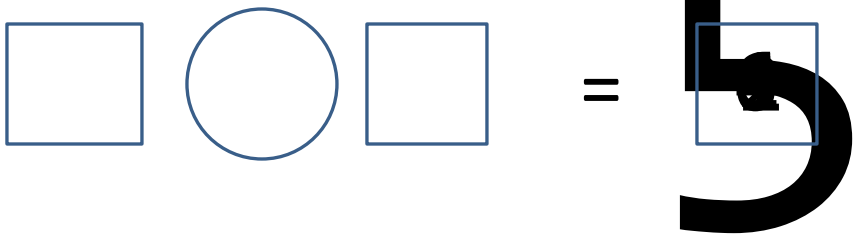
Пример

Входная строка:

```
5 2 3 * 4 2 / - 4 / + 1 -
```

Стек





Заключение

- Абстрактные типы данных
  - Несколько примеров
  - Определение
  - Зачем использовать АТД
- АТД список
  - Реализация на языке Си через 1-связные списки
- Другие АТД на основе списков
- Стек и примеры использования стеков
  - Перевод арифметического выражения из инфиксной в постфиксную запись
  - Вычисление значения выражения на стеке