Лекция 2

Представление данных (I).

Обзор. Очередь, стек, рекурсия.

2.1 Обзор

Переменная в компьютере действительно меняет значение. На самом деле, переменная — это место в памяти, в котором хранится текущее значение этой переменной.

Подпрограммы (процедуры, функции). Можно вызвать и вернуть управление на следующий оператор. Можно передать параметры.

Область действия переменной — подпрограмма (главная программа). Переменные при каждом вызове подпрограммы — это разные переменные (так же, как и параметры).

Структуры данных, которые мы изучим:

- Массив.
- Очередь.
- Стек.
- Файл.
- Списки (разные).
- Деревья (очень разные).
- Хеш-таблицы.
- ...

Структуры данных, в первую очередь, определяются *операциями* над ними, а также способом реализации.

Например, единственная легко реализуемая операция над *массивом* — `[·]` — получение доступа к элементу с заданным номером. Адрес, в котором хранится элемент, легко определить. Затем можно считать оттуда элемент или записать туда другой элемент. Например, на RAM-машине вычислить адрес требуемого элемента можно всего за одну операцию (сложения), его считывание или модификацию также можно проделать всего за одну операцию (с косвенной адресацией).

Напротив, для нахождения в *списке* элемента по его номеру придется проделать большое количество операций.

2.2 Очередь.

Операции: кладем в начало, вынимаем из конца (FIFO).
Реализация (возможная):

- массив (кольцо),
- указатели на первый занятый и первый свободный элемент

(*в этом случае* можно реализовать `[·]`).

2.3 Стек, рекурсия.

2.3.1 Рекурсивные процедуры.

Рекурсивной называется процедура, вызывающая себя. Вообще, вызов (любой) процедуры происходит так: во время исполнения программы с шага s происходит переход на адрес начала процедуры; вычисления продолжаются (заметим, что при рекурсивном вызове «одни и те же» локальные переменные в вызываемой и вызывающей копиях процедуры имеют разные значения); затем происходит возврат на шаг $s+1$.

Пример 2.1 (числа Фибоначчи).
function f (i : integer) : integer;
begin
    if $i = 0$ then return 0
    else if $i = 1$ then return 1
    else return $f(i-1) + f(i-2)$;
end; □

Пример 2.2 (задача о рюкзаке). Имеется N предметов и рюкзак объема V. Даны их объемы v_i и стоимости g_i. Требуется найти набор предметов максимальной стоимости, помещающийся в рюкзак.

function knapsack (V, N : integer, набор v_i и g_i) : набор целых чисел;
var *optG* : integer $= -1$;
    opt : набор целых чисел = пустой;
begin

```
    for k := 1 to N do   if v_k ⩽ V then
    begin
        next := {k} ∪ knapsack(V − v_k, N − k, набор при i ⩾ k + 1);
        nextG := стоимость(next);
        if (nextG > optG) then begin opt := next; optG := nextG; end;
    end;
    return opt;
end;
```
□

Пример 2.3 (проверка правильности выражения).
выражение ≡ сумма ;
сумма ≡ терм | терм + сумма
терм ≡ буква | (сумма)

Например, выражением является

`a+(b+(c+d)+e);`

Читаем входной поток функцией getnext : char. Процедура getback возвращает символ во входной поток (чтобы в следующий раз был прочтен тот же символ).

```
function expression : boolean;
begin
    if (not sum) return false;
    if (not getnext = ';') return false;
    return true;
end;

function sum : boolean;                    (* читает сумму до конца *)
begin
    if (not term) return false;
    if (not getnext = '+') begin getback; return true; end;
    if (not sum) return false;
    return true;
end;

function term : boolean;
begin
    if (getnext in ['a'..'z']) return true;

    if (not getnext = '(') return false;
    if (not sum) return false;
    if (not getnext = ')') return false;
```

```
    return true;
end;
```
□

2.3.2 Реализация рекурсии в компьютере: стек

Стек (LIFO):

Операции: PUSH, POP (и, если повезет с реализацией, `[·]`).
Реализация (стандартная):

- место в памяти (как массив),
- счетчик: верхушка стека,

Реализация рекурсии.

Вызов процедуры:

- PUSH адрес возврата.
- PUSH параметры.
- GOTO процедура.
- Передвинуть счетчик (PUSH 0) на размер памяти, необходимый для хранения локальных переменных.

Возврат:

- Передвинуть счетчик (POP) на размер памяти, в которой хранились локальные переменные и параметры.
- POP адрес возврата и GOTO туда.

Передача результата — зависит от реализации.

2.3.3 Избавление от рекурсии.

Способ 1: при помощи стека.

Реализовать стек в массиве.

Способ 2: динамическое программирование.

Пример 2.4 (числа Фибоначчи).
function g (j : integer, p : массив) : integer;
begin
    if $j = 0$ then return 0
    else if $j = 1$ then return 1
    else return $p[j-1] + p[j-2]$;
end;

function f (i : integer) : integer;
var a : массив;
begin
    for $j := 1$ to i do $a[j] := g(j, a)$;
    return $a[i]$;
end; □

Пример 2.5 (задача о рюкзаке).
Для всех i от 0 до V и k от N до 0 последовательно найдем оптимальный набор предметов с k-го до n-го, который можно уложить в рюкзак объема i (при этом можно пользоваться уже найденными оптимальными наборами для $i' < i$ и $k' > k$). □

Упражнение 2.1. Определить временнýю сложность в наихудшем случае всех алгоритмов, приведенных в разделе 2.3. □