

Лекция 1 (04.09.2018)

1.1 Рекурсия

Чтобы понять рекурсию, надо
понять рекурсию.

Рекурсивный алгоритм для вычисления вызывает себя.

Виды рекурсии:

1. Прямая рекурсия - функция А вызывает сама себя.
2. Косвенная рекурсия - А вызывает В, А вызывает В. В одной функции обязательно должна быть хотя бы одна нерекурсивная ветка

1.2 Процесс рекурсивного вычисления

Этапы процесса рекурсивного вычисления:

1. Погружение в рекурсию.
2. Всплывание из рекурсии.

1.3 Суперкосинус

$$\text{supercos}(x, n) = \underbrace{\cos \dots \cos}_n(x);$$

```
1 double supercos(double x, int n) {  
2     if (n == 0)  
3         return x;  
4     return cos(supercos(x, n-1));  
5 }
```

1.4 Алгоритм Евклида

Находит наибольший общий делитель чисел a и b .

```
1  int gcd(int a, int b) {  
2      if (b == 0) return a;  
3      return gcd(b, a % b);  
4  }
```

1.5 Функция Аккермана

Работает на множестве натуральных чисел.

$$A(m, n) = \begin{cases} n + 1, & m = 0 \\ A(m - 1, 1), & m > 0, n = 0 \\ A(m - 1, A(m, n - 1)), & m > 0, n > 0 \end{cases}$$

1.6 Разбор строки по грамматике

Нотационные формы Бэкуса-Наура

- $\langle \text{цифра} \rangle ::= 0|1|\dots|9$
- $\langle \text{буква} \rangle ::= a|\dots|z|A|\dots|Z|_$
- $\langle \text{идентификатор} \rangle ::= \langle \text{буква} \rangle | \langle \text{идентификатор} \rangle \langle \text{буква} \rangle | \langle \text{идентификатор} \rangle \langle \text{цифра} \rangle$
- $\langle \text{оператор} \rangle ::= \langle \text{выражение} \rangle | \langle \text{if} \rangle | \langle \text{while} \rangle | \dots$
- $\langle \text{while} \rangle ::= \text{while}(\langle \text{выражение} \rangle) \langle \text{оператор} \rangle$
- $\langle \text{формула} \rangle ::= \langle \text{цифра} \rangle | (\langle \text{формула} \rangle \langle \text{знак} \rangle \langle \text{формула} \rangle)$
- $\langle \text{знак} \rangle ::= +|-|^*$

Задачи:

1. Вычислить значение корректного выражения
2. Проверить корректность выражения.

Решаем первую задачу:

```
1  int form() {  
2      char c, z;  
3      int x, y;  
4      cin >> c;  
5      if (c >= '0' && c <= '9') return c - '0';  
6      x = form();  
7      cin >> z;  
8      y = form();  
9      cin >> c;  
10     if (z == '+') return x + y;  
11     if (z == '-') return x - y;  
12     if (z == '*') return x * y;  
13 }
```

Общие принципы вычисления выражений:

1. Для каждого понятия есть своя функция.
2. Если понятие нетривиальное - вызываем его функцию.
3. Тривиальные понятия анализируем явно.

Лекция 2 (11.09.2018)

2.1 Ханойские башни

```
1 void abc(int n, int a, int b, int c) {  
2     if (n > 0) {  
3         abc(n-1, a, c, b);  
4         cout << a << " --> " << b << endl;  
5         abc(n-1, c, b, a);  
6     }  
7 }
```

Общая идея: при $n = 1$ решение тривиально, а если мы умеем решать задачу для $n - 1$ дисков, то перенесём $n - 1$ дисков на третий стержень. Последний оставшийся диск перенесём на второй стержень. После этого перенесём все диски с третьего стержня на второй.

2.2 Быстрое возведение в степень

Неправильный код

```
1 int pow(int x, int n) {  
2     if (n == 0) return 1;  
3     if (n \% 2 == 0) {  
4         return pow(x, n / 2) * pow(x, n / 2);  
5     }  
6     return pow(x, n-1) * x;  
7 }
```

Недостатки кода: x нужно сделать глобальным, чтобы не передавать его в рекурсии, т.к. он не изменяется. Также такой код совершает $O(N)$ итераций.

Правильный код

```
1  int x;  
2  int pow(int n) {  
3      if (n == 0) return 1;  
4      if (n \% 2 == 0) {  
5          int m = pow(n / 2);  
6          return m*m;  
7      }  
8      return x * pow(n - 1);  
9  }
```