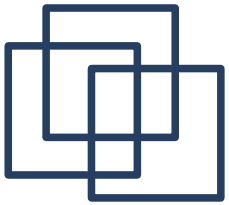
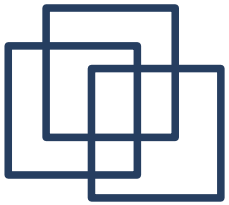


Кафедральный практикум 5 семестр

Часть 1. Введение в функциональное программирование (заключение)

http://sp.cmc.msu.ru/~kornykhin/fp/slides/part1-3.pdf



План

Часть 1.

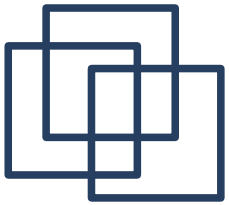
1) Организационные вопросы

2) Функциональный стиль программирования

Часть 2. Язык программирования Scheme.

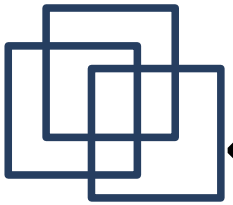
Часть 3. Функции высшего порядка. «Векторное» мышление.

Часть 4. Теоретический фундамент ФП.



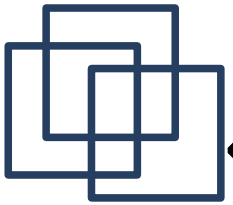
Умникам и умницам

- see "advanced" at http://sp.cmc.msu.ru/~kornykhin



«Expression only» стиль

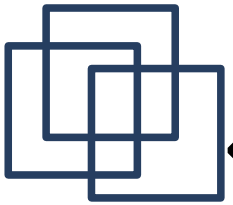
- «Expression only» - значит, «без statement'ов», за исключением return.
- В простейшем случае, тело функции — return с выражением.
- Разрешаем использовать константные локальные переменные.
- Разрешаем использовать «if-then-else», но при условии, что он ведет себя как «? :».



«Expression only» стиль

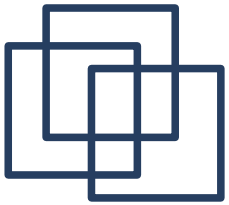
- Тело_функции ::=
  {
    {Объявление_лок_перем}
    (**return** выражение **;**
    | **if (**выражение**)** Тело_функции
                else Тело_функции
)
  }

- Объявление_лок_перем ::=
    const Тип Имя **=** выражение **;**



«Expression only» стиль

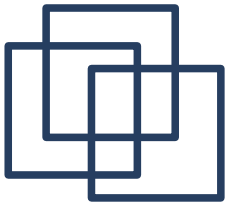
- Тело_функции ::= Возврат | Блок
- Возврат ::= **return** выражение **;**
  | **if (**выражение**)** Тело_функции
  else Тело_функции
- Блок ::=
  {
  {Объявление_лок_перем}
  Возврат
  }
- Объявление_лок_перем ::=
  const Тип Имя **=** выражение **;**



Работа с «кучей»

- Все указатели должны быть **const T * const**
- Нужен struct → пишем в нем конструктор, в котором инициализируем поля структуры.
- Выделение памяти при помощи **new**.
- Для списков можно использовать fplists

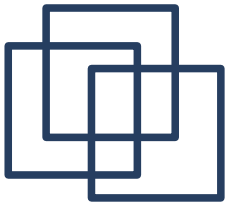
```
int amount(const struct tree * const t)
{
   if (!t) return 0;
   else return amount(t->left) +
               amount(t->right) + 1;
}
```



Задача

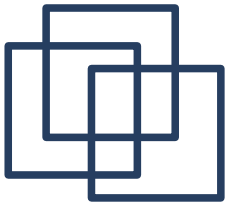
- Описать структуру двоичного дерева tree
- В стиле expression-only реализовать добавление элемента в дерево:

```
const struct tree * const
insert ( const struct tree * const source
       , const int value  )
{
    ....................
}
```



Максимум в списке

```
#include "fplists.h"
int max(const list xs) {
  return max_tail(xs, 0);
}
int max(const int a, const int b) {
  return a > b ? a : b;
}

int max_tail(const list xs, const int cmax) {
   if (len(xs) == 0) return cmax;
   else return max_tail(tail(xs),
                        max(first(xs), cmax) );
}
```

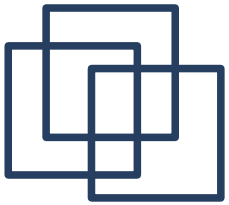


«Хвостовая» рекурсия

- Все return имеют вид:
  - **return expr**;
  - ИЛИ **return f(expr1, ..., exprN);**

    (expr, expr1, ..., exprN - без вызовов рекурсивных функций, f — рекурсивная функция)

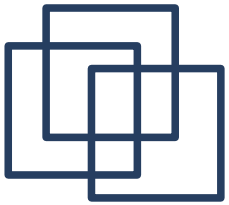
- Вызовы рекурсивных функций не встречаются в условиях if-then-else и при объявлении локальных переменных
- Всё то же и про функции, *приводящие к рекурсивным функциям.*



Взаимная рекурсия

```
int f(const int x) {
  if (g(x) > 0)
     return f(x — 1);
  else
     return g(x);
}


int g(const int x) {
    return f(x/2);
}
```

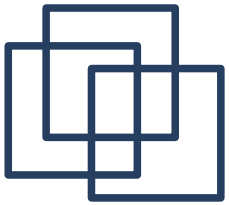
- Тут рекурсия хвостовая?



Взаимная рекурсия

```
int f(const int x) {
  if (x < 2)
     return f(x — 1);
  else {
     const int y = g(x);
     return y + 1;
   }
}

int g(const int x) {
    return f(x/2);

}
```

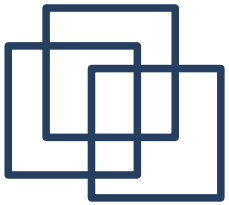
- Тут рекурсия хвостовая?



Взаимная рекурсия

```
int f(const int x) {
  if (x < 2)
     return f(x — 1);
  else {
     const int y = g(x);
     return y;
   }
}

int g(const int x) {
    return f(x/2);

}
```

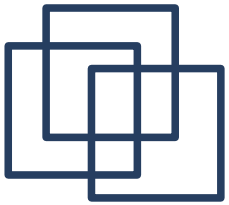
- Тут рекурсия хвостовая?



Взаимная рекурсия

```
int f(const int x) {
  if (x < 2)
    return f(x — 1);
  else
    return g(x);
}

int g(const int x) {
   return f(x/2);
}
```

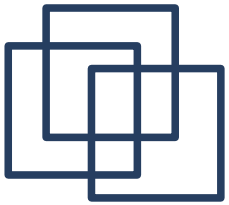
```
int m(const int x) {
  int _x = x;
  f:
   if (_x < 2) {
     _x = _x — 1;goto f;
   } else {
     goto g;
   }

  g:
     _x = _x/2; goto f;
}
```



Взаимная рекурсия

- **Семантическое определение хвостовой рекурсии** — это такая рекурсия, которую можно преобразовать в "goto с метками и локальными переменными", как на картинке справа.

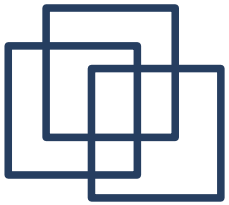
```
int m(const int x) {
  int _x = x;
  f:
   if (_x < 2) {
      _x = _x — 1;goto f;
   } else {
      goto g;
   }

  g:
      _x = _x/2; goto f;
}
```



Аккумулятор

- Вопрос: какая общая идея "мышления" в аккумуляторном стиле?
  - на какие вопросы надо ответить?
  - какой общий вид функции с аккумулятором?

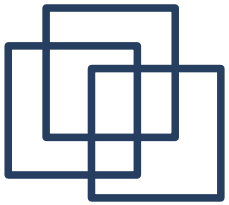
- NB: Раз рекурсия хвостовая, то списки можно просматривать только от начала к концу (последовательно), а формировать новые списки, только лишь приписывая новые элементы к голове списка!



Аргументы-"состояние"

- изменение содержимого фиксированной области памяти
- Пример: `while (cond) { d ← updated(d); }` преобразуется в

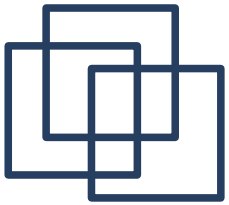
```
int loop( int d )
{
    if (cond)
      return loop(updated(d));
    else
      ...;
}
```



Задача

- Переписать с хвостовой рекурсией:

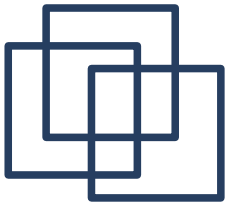
```
int fibonacci(const int n)
{
    if (n < 2) return n;
    else return fibonacci(n-1) +
                fibonacci(n-2);
}
```



Упражнение

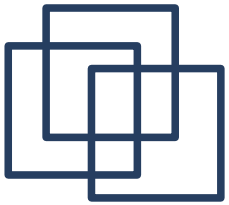
- Как переписать программу, чтобы функция h реализовала хвостовую рекурсию ? f, g — независимые рекурсивные функции, уже реализованные с хвостовой рекурсией

```
list f(const list xs);

list g(const list xs);

list h(const list xs) {
    return f(g(xs));
}
```



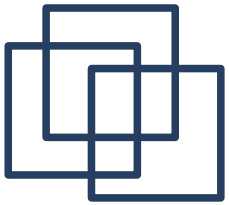
fplists

- Нерекурсивные функции в fplists :
  - first
  - second
  - tail
  - is_empty
  - create
- Рекурсивные функции в fplists :
  - length



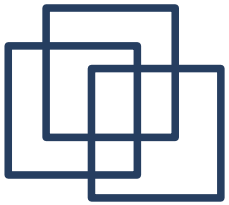
Задачи

- Решить (вся рекурсия должна быть хвостовой):
  - "Удвоение": из данного списка (н-р, 1-2-3) постройте его же с копией на конце (т. е. 1-2-3-1-2-3);
  - "Длинная голова": из данного списка (н-р, 1-2-3) и числа N от 1 до длины списка (н-р, 2) оставьте от списка первые N элементов (н-р, 1-2);
  - "Бесповторность": дан список, проверить, есть ли в ней дубликаты.



Анонс

- На следующем семинаре будем разбираться с Scheme, в том числе в техническом плане.
- Хотите сразу всё установить к себе на ноутбук, решив все возникшие при этом проблемы на семинаре? :) **Возьмите свой ноутбук с собой на семинар!**



Домашнее задание

1) Реализовать в стиле «из одних выражений» несколько функций на C++ (**вся рекурсия должна быть хвостовой**)
   - нерешенные задачи с семинара
   - http://sp.cmc.msu.ru/~kornykhin → hw #3.1
2) Решения оформлять в виде текстовых файлов и присылать мне на почту kornevgen@cmc.msu.ru
3) Подумать, что дает такой стиль написания программ для качества программ, дает ли новую парадигму программирования