АНАЛИТИЧЕСКИЙ ПОДХОД К ДОВУЗОВСКОМУ ПРЕПОДАВАНИЮ ПРОГРАММИРОВАНИЯ

Илья Дединский
кафедра информатики МФТИ
http://ded32.ru

**Программисты в России часто растут «как трава».
Ее никто толком не выращивает, она «сама» пробивается сквозь асфальт и камни, ЕГЭ и олимпиады.
Хорошо ли это?**

АНАЛИТИЧЕСКИЙ ПОДХОД К ДОВУЗОВСКОМУ ПРЕПОДАВАНИЮ ПРОГРАММИРОВАНИЯ

Илья Дединский
кафедра информатики МФТИ
http://ded32.ru

Цель

Разработка практико-ориентированной методики довузовского преподавания программирования, полностью достаточной для успешного дальнейшего развития и саморазвития ученика в условиях современного ВУЗа и современного состояния индустрии разработки ПО.

Основные задачи

- Способы определения и преодоления образовательных разрывов «школа – ВУЗ – профессия»
- Выделение базовых принципов проектного обучения программированию, построение его последовательности и разработка основных курсов для школьного периода

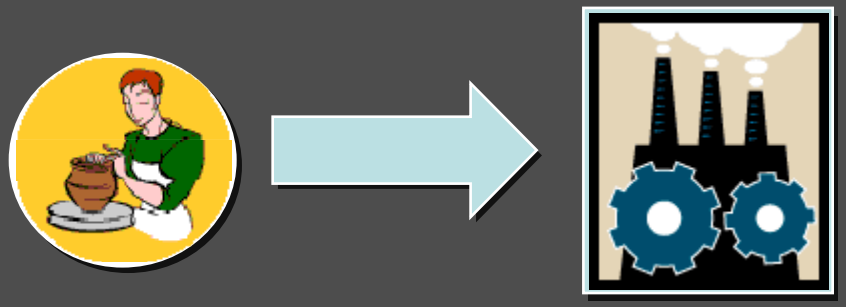
1. МОТИВАЦИЯ

2. МЕТОДИКА

3. РЕЗУЛЬТАТЫ

ПРОГРАММИРОВАНИЕ – ТРИ В ОДНОМ

ЧЕРТЫ ИНДУСТРИИ – МЕЙНСТРИМ



Что было раньше? – кустарная модель

- Научный стиль разработки (ориентация на метод)
- Сотрудничество с заказчиком
- Пользование компьютером предполагало навыки программирования
- Меньший объем проектов
- Локальная разработка
- Преимущественно индивидуальная разработка
- Небольшое количество стандартов
- Небольшой объем литературы

- Культура программирования только развивалась
- **Легко было учиться самому**

Что сейчас? – промышленная модель

- Промышленный стиль разработки (ориентация на продукт)
- Согласование с заказчиком
- Документирование
- Ограничения по срокам
- Большой объем
- Коллективность
- Много стандартов, инструментов
- Чужой и устаревший код
- Удаленная разработка
- Большой объем литературы

- Без культуры программирования работать невозможно!
- **Учиться самому все труднее и труднее**

- В результате без опыта промышленной разработки даже самая светлая голова – убытки для компании.

ВУЗОВСКОЕ ОБРАЗОВАНИЕ – МЕЙНСТРИМ

Что сейчас? (МФТИ, ВМК МГУ – отчасти счастливые исключения)

- Как раздел математики, либо офисные технологии
- Отрыв обучения от индустрии
- Не усваиваемые порядок и темп изложения
- Практика: фрагментарность, ориентация «на методичку». И – мало практики!
- Нет командных технологий
- Нет рефлексии – не «ставится голова»
- Трудно приобрести нужные навыки самому
- Это скорее отбор **уже** успешных, а не обучение одаренных.
  Оно больше соответствует прошлому периоду развития индустрии, чем нынешнему.

Исключения, ставшие мейнстримом:

- Олимпиады
- В какой-то степени – курсовые и дипломные работы, спецкурсы, доп. образование (МФТИ: Intel, Parallels, NetCracker и др.)

Как быть студенту?

- Заниматься олимпиадами (но – до определенной стадии)
- Самому идти на кафедру, брать тему и выбиваться в люди
- Оставить надежду на ВУЗ, учиться самому, идти работать и выбиваться в люди

- Вообще, никакая система обучения не гарантирует порождения Специалиста.
  Она может этому только способствовать.
- Кроме того, надо искать соответствующие ВУЗы. Но сколько их?

ШКОЛЬНОЕ ОБРАЗОВАНИЕ – МЕЙНСТРИМ

То же, что и в вузе, плюс:

- Великая ориентация на пользование ПК, иногда – на олимпиады
- Очень разный и часто низкий уровень подготовки учителей
- Разнородное, неустойчивое и утилитарное мнение родителей, администрации школ (программист – это вид сантехника)
- Такая же мотивация учеников (скачать весь Интернет… взломать сервер Microsoft…)
- Слабое сопряжение со смежными предметами (у всех свои заботы…)
- Низкий уровень проектных работ (проекты по **истории** программирования вместо собственно программирования, скачанные из Интернета рефераты…)
- Отношение администрации (программирование – не экзамен, **не** основной предмет)

Исключения, ставшие мейнстримом:

- Олимпиады
- Дополнительное образование (но здесь многие пед. технологии теряют устойчивость).
- Самостоятельное изучение, но это все труднее и труднее, несмотря на книги и Интернет

Как быть школьнику?

- Заниматься олимпиадами и надеяться, что в вузе и **на работе** будет то же самое.
- Самому искать научника, брать тему и выбиваться в люди. Но это труднее, чем студенту.
- Оставить надежду на школу, учиться самому и выбиваться в люди. Так многие и делают.
- Искать соответствующие школы. Но сколько их?

В результате программисты в России растут «как трава». Ее никто не выращивает, она «сама» пробивается сквозь асфальт и камни. Хорошо ли это?

1. МОТИВАЦИЯ

2. МЕТОДИКА

3. РЕЗУЛЬТАТЫ

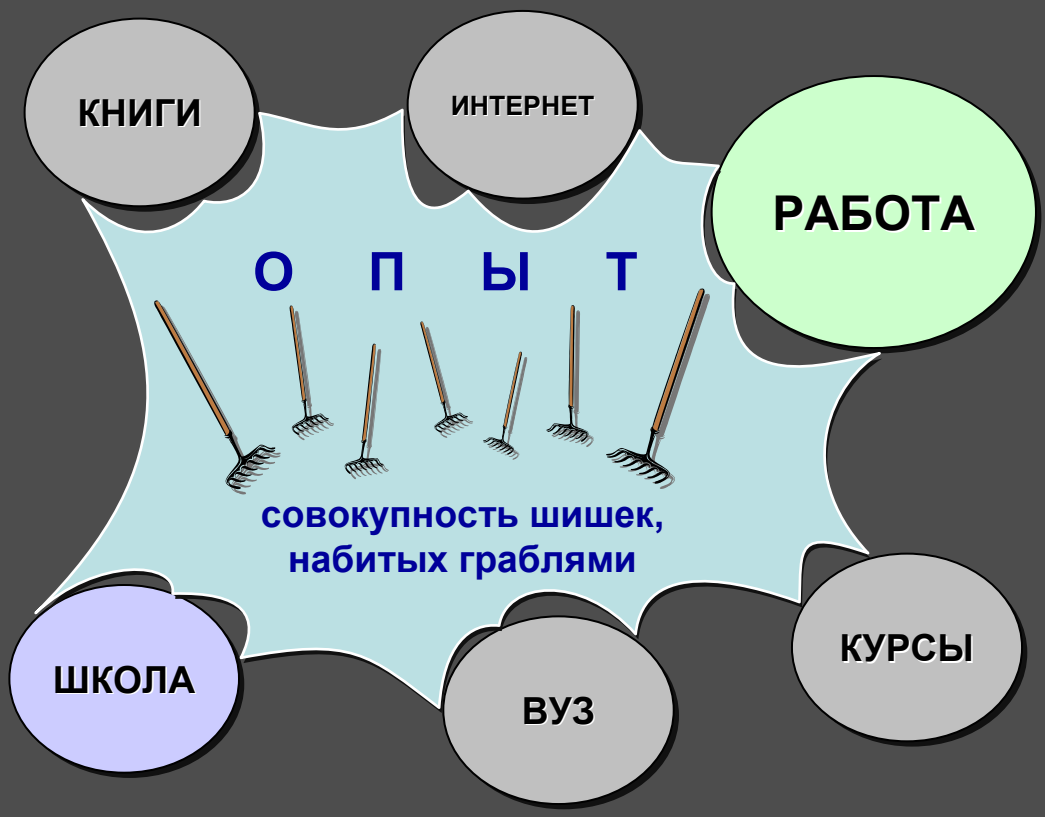
ОБРАЗОВАТЕЛЬНЫЕ РАЗРЫВЫ

Способы преодоления
- Самостоятельный метод
- Олимпиадный метод
- Непрерывное профильное образование

Повышение КПД образования
- Использование ВУЗа как ресурса

Для этого нужны:
- Активная учебная позиция
  - Мотивация
  - Конструктивное мышление
- Результаты и наработки

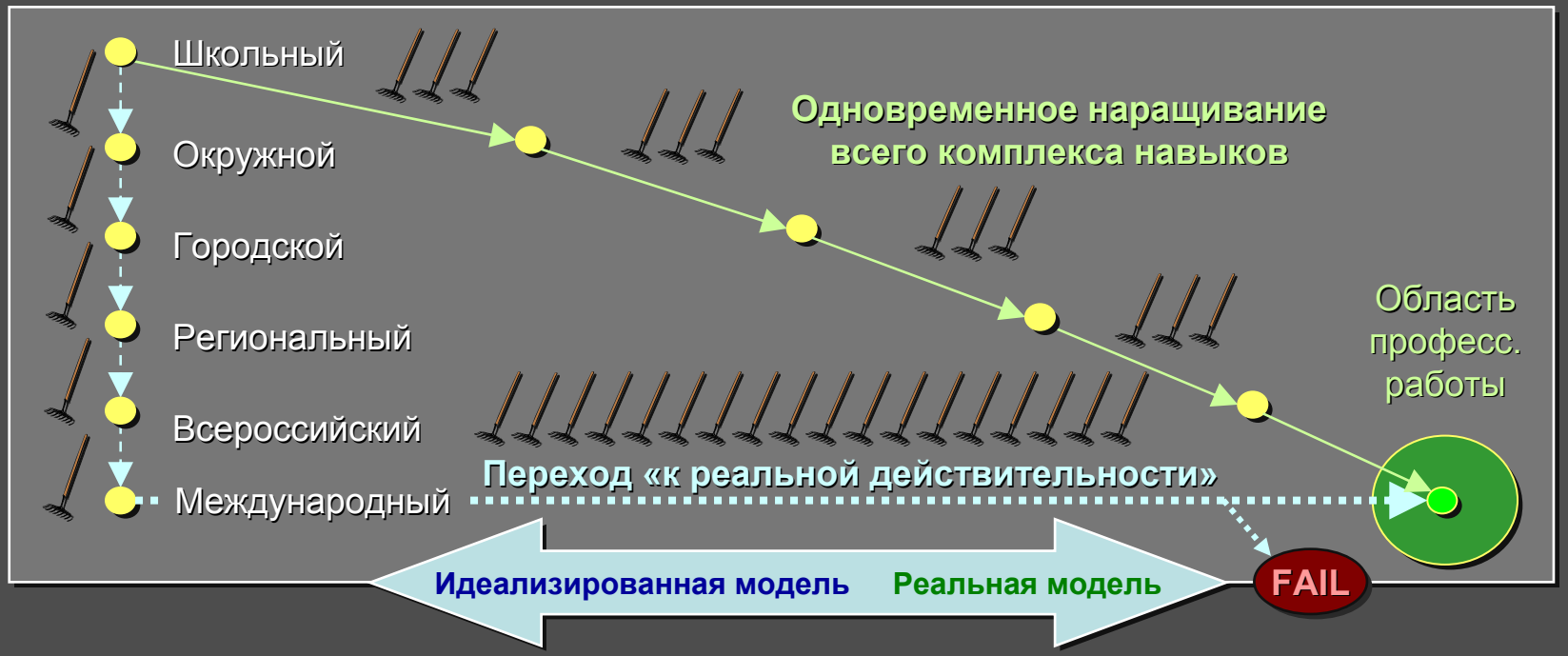
Следствие:
- Это необходимо приобрести до ВУЗа!



Печальный вывод:
- Без грамотно поставленной работы в школе первокурснику будет очень трудно. От лучших сокурсников он зачастую отстал навсегда.

Конструктивный вывод:
- Нужна методика, закладывающая фундамент и облегчающая «пользование ВУЗом как ресурсом»
- Эта методика должна и «ставить голову» и давать конкретную отдачу в виде материалов, которые можно показать будущим преподавателям и научным руководителям.

МОДЕЛИ ОПЫТА



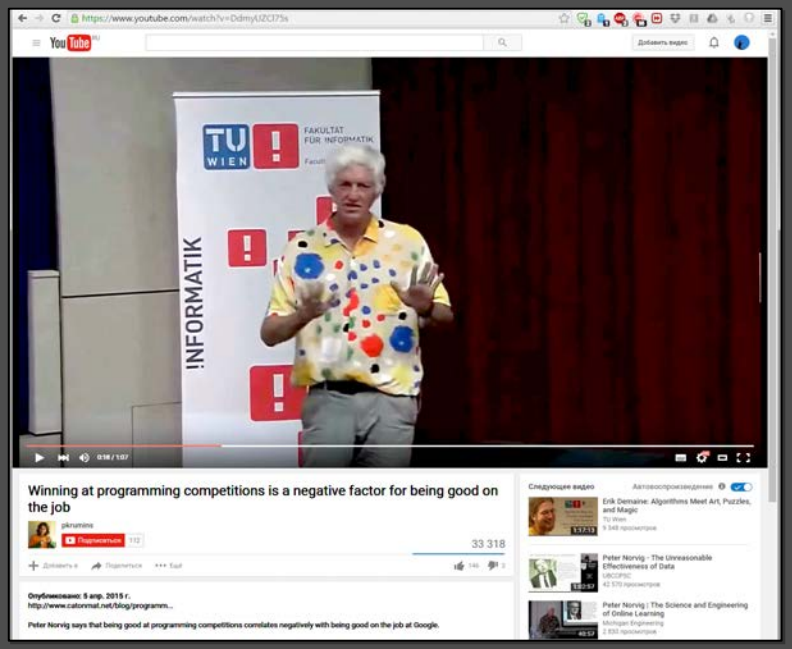
Локальную абстрактную задачу	Ориентация на	Целостный продукт
Небольшой	Объем задачи	Большой
Узкий и постоянный	Контекст задачи	Широкий и меняющийся
Импульсная фрагментарная	Характер работы	Непрерывная итерационная
Только со своим	Работа с кодом	Со своим и чужим
Индивидуальная	Разработка	Командная
Мало	Стандарты и литература	Много
Нет	Документация и сопровождение	Обязательны

Пед. задачи: внутренние (поэтому легко)

Пед. задачи: ВНЕШНИЕ (поэтому трудно)

Требуется методика, максимизирующая набор именно опыта. Все остальное придет и так.

МОДЕЛИ ОПЫТА



«Это было удивительно для меня, но победы в конкурсах по программированию негативно коррелируют с успехами в работе»

Питер Норвиг (Peter Norvig),
директор по исследованиям Google,
советник Ассоциации по улучшению искусственного интеллекта,
автор одного из самых популярных вузовских учебников по ИИ

Локальную абстрактную задачу	Ориентация на	Целостный продукт
Небольшой	Объем задачи	Большой
Узкий и постоянный	Контекст задачи	Широкий и меняющийся
Импульсная фрагментарная	Характер работы	Непрерывная итерационная
Только со своим	Работа с кодом	Со своим и чужим
Индивидуальная	Разработка	Командная
Мало	Стандарты и литература	Много
Нет	Документация и сопровождение	Обязательны

Требуется методика, максимизирующая набор именно опыта. Все остальное придет и так.

ОБУЧЕНИЕ ЧЕРЕЗ РЕФАКТОРИНГ

Что такое «знать»?
Когнитивно-технологическая единица (КТЕ)

- Зачем это надо?
- Что это такое?
- Где это можно и где нельзя использовать?
- Как это применять?
- На чем основано и с чем связано?
- Чем придется пожертвовать?
- Что будет, если этого не делать?
- Какие в этом «подводные камни» (чего опасаться и «накладные расходы»)

КОМПЬЮТЕРРА ONLINE

Как хотеть учиться

Автор: Илья Дединский
Опубликовано в журнале "Компьютерра" №24 от 23 июня 2005 года

По моему мнению, главная задача вуза, и не только в сфере ИТ, - научить студента действовать грамотно и самостоятельно…

«ТЕХНОЛОГИЯ ГРАБЛЕЙ»: МОДЕЛИРОВАНИЕ СИТУАЦИИ

…с иронией и юмором

Технология подается максимально «вовремя», на выверенном примере.

ПРИМЕР РЕФАКТОРИНГА

Второе занятие начального курса: Выделение функций. Первый рефакторинг.

- Чему учит: Структурирование, Культура именования
- Сразу – довольно большие программы (200-300, а то и до 1000 строк). Сразу – пишем аккуратно.

```
#include<TXLib.h>
  int main()
{txCreateWindow(630,390);
   txSetColor(TX_WHITE);
 txLine(90 ,80,180 ,80);
txLine(180,80,180,130);
   txLine(180,130,70,130);
 txLine(70,130,70,80);
txLine(70,130 ,120,180);
   txLine(70,180,120 ,130 );
txLine(120  ,130,180,180);
   txLine(180,130 ,130,180);
txSetColor(TX_RED);//vrode
domik?
 txLine (460,230 ,480,200);
txLine(580,200,610,200);
txLine(610,200,610,250);
 txLine (610, 200,590,220);
txLine(590,220,460,230);
 txLine(460, 230,460,280);
 txLine (490,230, 490,280);
txSetColor(TX_BROWN);
txLine(270  ,380,270,280);
 txLine (270,280,340,280);
txLine(340 ,280,340,380);
txCircle(380, 330,2);
return 0;}
```

«This code smells» M. Fowler

Противно

```
#include <TXLib.h>

int main ()
{
txCreateWindow (630, 390);
txSetColor (TX_WHITE);

txLine ( 90,  80, 180,  80);
txLine (180,  80, 180, 130);
txLine (180, 130,  70, 130);
txLine ( 70, 130,  70,  80);
txLine ( 70, 130, 120, 180);
txLine ( 70, 180, 120, 130);
txLine (130, 130, 180, 180);
txLine (180, 130, 130, 180);

txSetColor (TX_GREEN);
txLine (460, 230, 480, 200);
txLine (580, 200, 610, 200);
txLine (610, 200, 610, 250);
txLine (610, 200, 590, 220);
txLine (590, 220, 460, 230);
txLine (460, 230, 460, 280);
txLine (490, 230, 490, 280);

txSetColor (TX_BROWN);
txLine (270, 380, 270, 280);
txLine (270, 280, 340, 280);
txLine (340, 280, 340, 380);
txCircle (380, 330, 2);

return 0;
}
```

Утомительно

```
#include <TXLib.h>

void HouseDraw();
void TreeDraw();

int main ()
{
txCreateWindow (630, 390);

HouseDraw();
TreeDraw();
DogDraw();

return 0;
}

void HouseDraw()
{
txLine ( 90,  80, 180,  80);
txLine (180,  80, 180, 130);
txLine (180, 130,  70, 130);
txLine ( 70, 130,  70,  80);
txLine ( 70, 130, 120, 180);
txLine ( 70, 180, 120, 130);
txLine (130, 130, 180, 180);
txLine (180, 130, 130, 180);
}

void Tree::Draw()
{
txSetColor (TX_GREEN);
txLine (460, 230, 480, 200);
```

Удобно

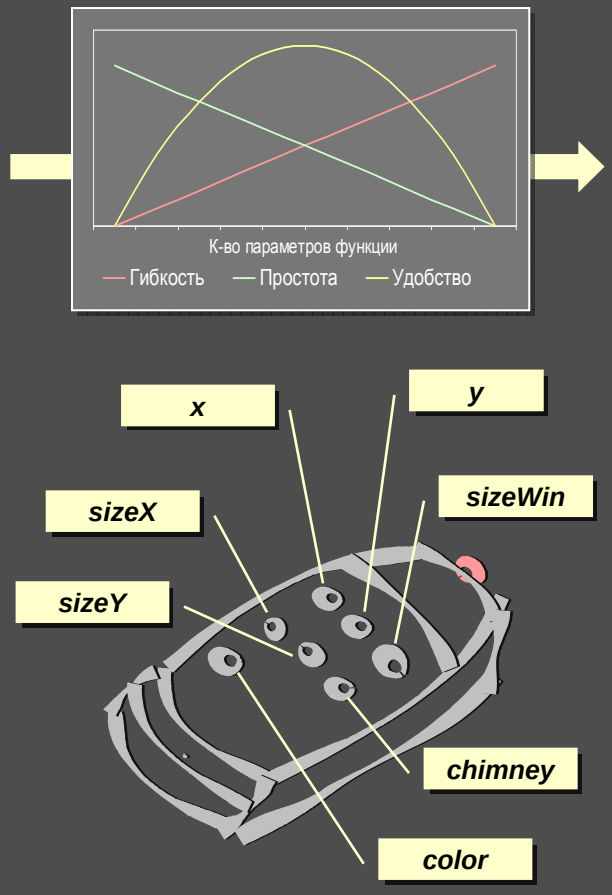
ПРИМЕР РЕФАКТОРИНГА

Третье занятие начального курса: Функции с параметрами

- Функция как черный ящик (неуправляемый – без параметров, управляемый – с параметрами)
- Первый опыт (зародыш) построения API и контрактного программирования

```
#include <TXLib.h>

void HouseDraw ();
void TreeDraw();

int main ()
{
txCreateWindow (630, 390);

HouseDraw ();
TreeDraw();
DogDraw();

return 0;
}

void HouseDraw ()
{
txLine (  90,  80, 180,   80);
txLine (180,  80, 180,  130);
txLine (180, 130,  70,  130);
txLine (  70, 130,  70,   80);
txLine (  70, 130, 120, 180);
txLine (  70, 180, 120, 130);
txLine (130, 130, 180, 180);
txLine (180, 130, 130, 180);
}

void TreeDraw()
{
txSetColor (TX_GREEN);
txLine (460, 230, 480, 200);
…
```

Сложно управлять
(надо пересчитывать координаты)



Набор параметров хорошо показывает понимание темы

```
#include <TXLib.h>

void HouseDraw (int x, int y,
                int sizeX, int sizeY,
                int sizeWin, int chimney,
                COLORREF color);
…
int main ()
{
txCreateWindow (630, 390);

HouseDraw (100, 150, 100, 40, 20, …);
TreeDraw (…);
DogDraw (…);

return 0;
}

void HouseDraw (int x, int y,
                int sizeX, int sizeY,
                int sizeWin, int chimney,
                COLORREF color)
{
txSetColor (color);
txLine (x – sizeX/2,  y + sizeY,
        x + sizeX/2,  y + sizeY);
…
}

void TreeDraw (int x, int y, int size,
               int branchesSize,
               int branchesUp,
               …)
```

Легко управлять
(надо лишь задать параметры)

ФОРМИРОВАНИЕ СИСТЕМЫ ЦЕННОСТЕЙ

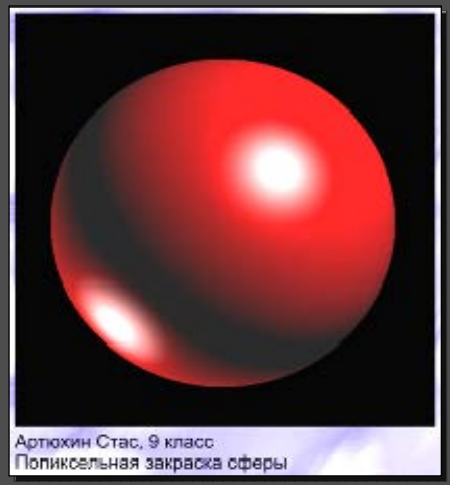
№	Тема / ученик должен знать / ученик должен уметь (обучение)	Ученик должен чувствовать (воспитание)
5	Смысловое разделение частей алгоритма. Концепция процедурного программирования. Функции без параметров. Оформление функций. Вызов функции. Пошаговая отладка в программе с функциями. Именование функций. Понятие стиля программирования.	Понимать, что функции облегчают программирование и понимание программы. Правильно разделять работу между функциями. Радоваться созданию новой функции. Называть функции по смыслу, понимать роль «говорящих» имен. Не любить длинных функций.
6	Практика: рефакторинг проекта «мультфильм» с применением функций без параметров.	Относиться к функции как к инструменту, который должен быть удобным и безопасным. Не писать функций «с медвежьей услугой».
7	Функции с параметрами, возможности, которые они предоставляют. Грамотное использование параметров функции (взаимная независимость). Числовой тип данных. Отладка программы с переменными и параметрами. Понятие о библиотеке функций.	Уметь гибко «настраивать» функцию с помощью параметров, не любить негибкие функции. Уметь искать «золотую середину» при составлении набора параметров. Правильно называть параметры, любить «говорящие» имена. Группировать функции в библиотеки по смыслу. Понимать, как библиотеки облегчают программирование и любить их использовать.
8	Практика: рефакторинг проекта «мультфильм» с применением функций с параметрами.	
9	Повторяющиеся действия в алгоритмах. Разбор циклических алгоритмов. Цикл while. Работа с переменными в цикле. Отладка программы с циклами. Ошибки при работе с циклами.	Понимать, что циклы облегчают программирование и делают его гораздо более интересным. Опасаться зацикливания и всегда контролировать триаду «начальное значение – изменение – конечное значение». Использовать отладчик и распечатку значений на экран для понимания выполнения, поиска ошибок (и иметь опыт этого поиска).
10	Практика: доработка проекта «мультфильм» с использованием циклических алгоритмов.	

Сформулирована ценностная компонента (метацель) занятий, в терминах «любви» и «ненависти»

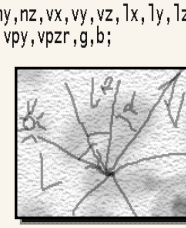
ВОСПИТАНИЕ ТЕХНОЛОГИЧЕСКОЙ КУЛЬТУРЫ

Выработка системы ценностей и критериев качества

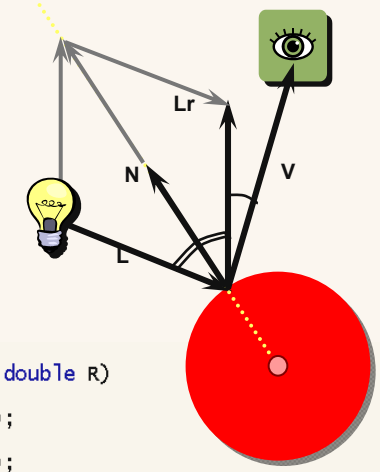
- Ясность
  Выразительность
- Надежность
  Сопровождаемость
- Модульность
  Архитектурная логичность
- Масштабируемость
- Поддержка стандартов
  Переносимость
- Навыки командной работы
- **Культура как контекст**




```
int main()
{
double t,j,i,px,py,pz,q,nx,ny,nz,vx,vy,vz,lx,ly,lz;
double diff,lpx,lpy,lpz,vpx,vpy,vpzr,g,b;
vpx=vpy=0;
vpz=500;
for(t=3.14/2;;t+=0.1)
for(j=-100;j<=100;j++)
for (i=-100;i<=100;i++) {
if (i*i+j*j>10000) continue;
lpx=-200*(1+cos(t));
lpy=-200*cos(t);
lpz=200*sin(t);
px=i; py=j; pz=sqrt(10000-i*i-j*j));
q=sqrt(px*px+py*py+pz*pz);
nx=px/q,ny=py/q,nz=pz/q;
q=sqrt((vpx-px)*(vpx-px)+(wpy-py)*(wpy-py)+(wpz-pz)*(wpz-pz));
vx=(wpx-px)/q; vy=(wpy-py)/q;vz=(wpz-pz)/q;
q=sqrt((lpx-px)*(lpx-px)+(lpy-py)*(lpy-py)+(lpz-pz)*(lpz-pz));
vx=(lpx-px)/q; vy=(lpy-py)/q; vz=(lpz-pz)/q;
diff=vx*lx+vy*ly+vz*lz; if(diff<0) diff=0;
r=b=0.2;
g=0.07*diff+0.2;
if(r>1)r=1; if(r<0)r=0; if(g>1)g=1; if(g<0)g=0; if(b>1)b=1; if
SetPixel(hwnd,i+100,j+100,RGB(r*255,g*255,b*255));
}
}
```



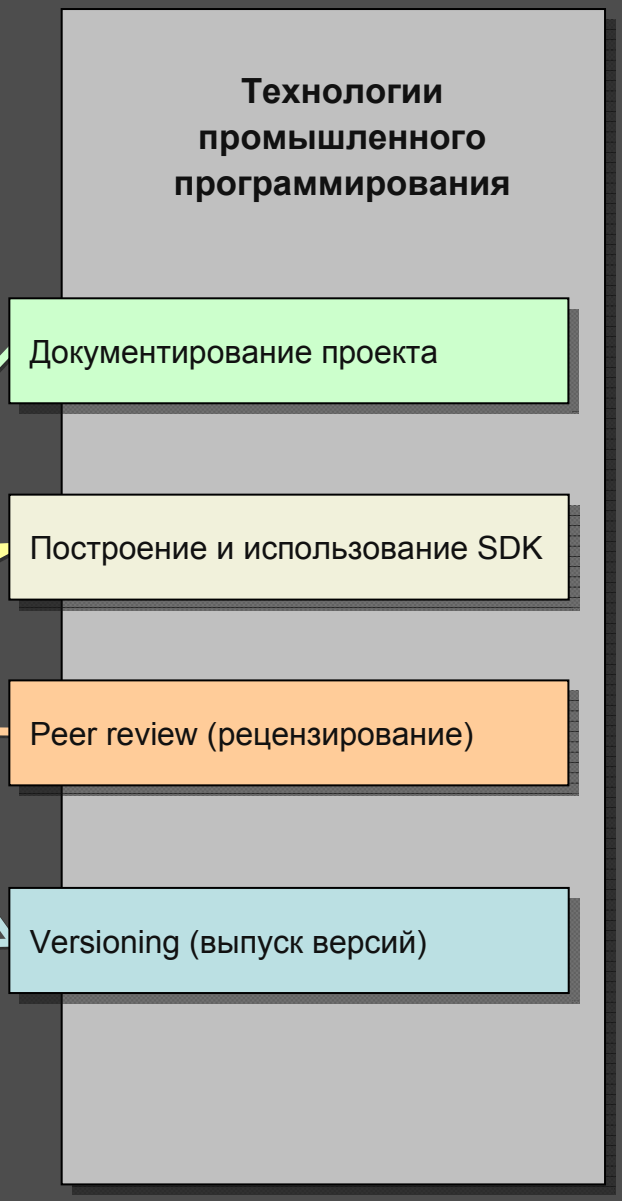
```
//--------------------------------

int main()
    {
    double R = 100;

    txCreateWindow (2*R, 2*R);

    for (double t = txPI/2; ; t +=
        DrawScene (vec (-2*R * (1 +
                        -2*R * cos
                        +2*R * sin
    return 0;
    }

//--------------------------------

void DrawScene (const vec& lightPos, double R)
    {
    vec viewPos       (  0,  0, +5*R);

    vec materialColor (0.0, 1.0, 0.0);
    vec lightColor    (1.0, 0.7, 0.0);
    vec ambientColor  (0.2, 0.2, 0.2);

    for (double y = -R; y <= R; y++)
        for (double x = -R; x <= R; x++)
            {
            if (x*x + y*y > R*R) continue;

            vec p (x, y, sqrt (R*R - x*x - y*y));
            vec N = !p;

            vec V = ! (viewPos  - p);
            vec L = ! (lightPos - p);

            double diffuse = N ^ L;
            if (diffuse < 0) diffuse = 0;

            vec Lr = 2*(N^L)*N - L;
            double spec = Lr ^ V;
            if (spec < 0) spec = 0;
            double specular = pow (spec, 25);

            vec color = ambientColor * materialColor +
                        diffuse      * materialColor * lightColor +
                        specular     * lightColor);

            DrawPixel (x+R, y+R, color);
            }
    }
```



ВОСПИТАНИЕ ТЕХНОЛОГИЧЕСКОЙ КУЛЬТУРЫ

Промышленные технологии уже в первом проекте

- Выделение модуля (модулей) для разработки мультфильма
- Библиотеки (на уровне include-файлов с кодом. Это не совсем профессионально, но в раздельной компиляции 7(8)-классники путаются).
- Деление функции на «сценарные» (вспомогательные для основной программы), «библиотечные» (API модуля) и «вспомогательные для библиотеки» (в терминах раздельной компиляции на C они соответствуют статическим функциям).
- Документация по библиотеке с героями мультика (doxygen).
- Выпуск mini-SDK (code + doc + example). Versioning.
- Перекрестный обмен кодом, задание – разобраться в чужом SDK и написать на нем нано-мультик за 1-2 урока.
- Домашнее задание – (перекрестная) рецензия на использованный SDK (code/doc peer review) и рефакторинг своей библиотеки с учетом этой рецензии. Отдельная «песня» – научиться писать рецензии.
- Надо сдать много чего: наномультик, библиотеку, рецензию, переработанную библиотеку.
- На эту работу удобно потом часто ссылаться, в ней много знаковых моментов.

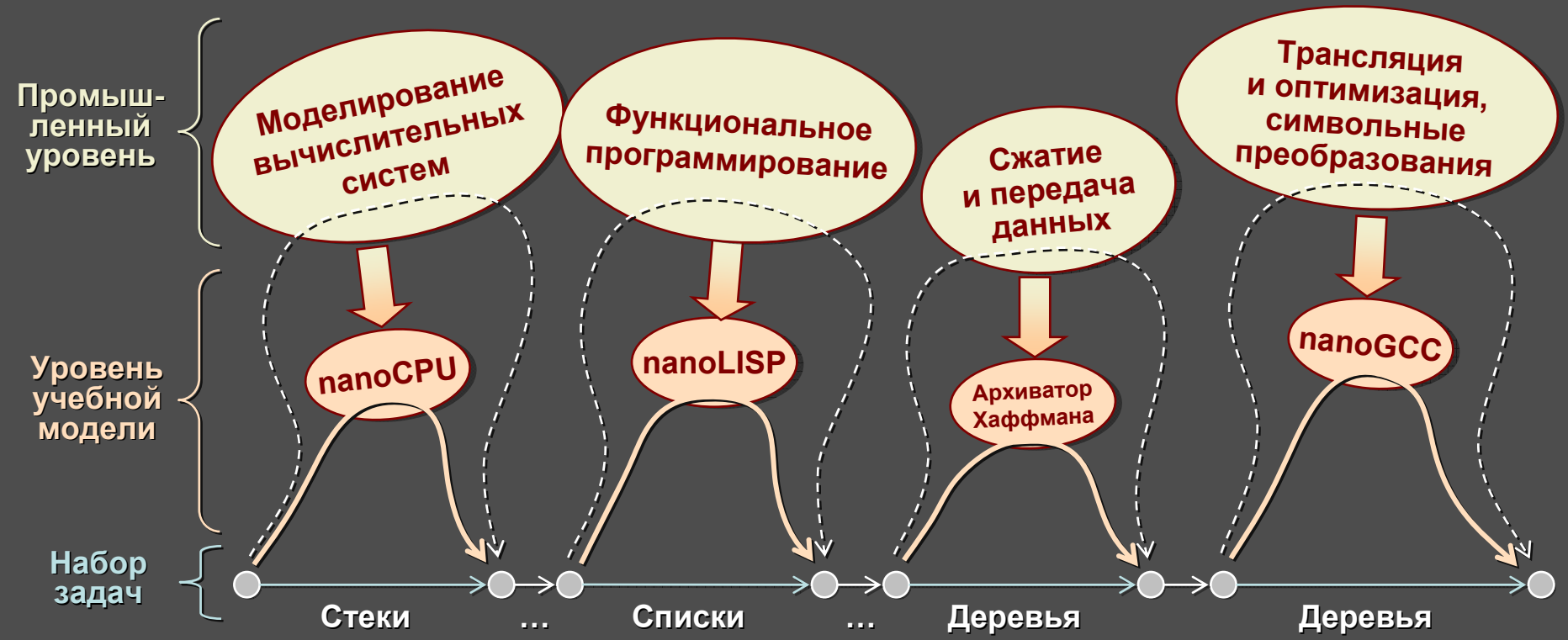


Кто ясно мыслит – ясно излагает.
Обратное часто тоже верно.

ПОРОЖДЕНИЕ ПРОЕКТНЫХ ЭЛЕМЕНТОВ

Пример: курс «Структуры данных и алгоритмы»
Третий год обучения (9 или 10 класс)

(Опущены: очереди, хеш-таблицы, графы, автоматы)

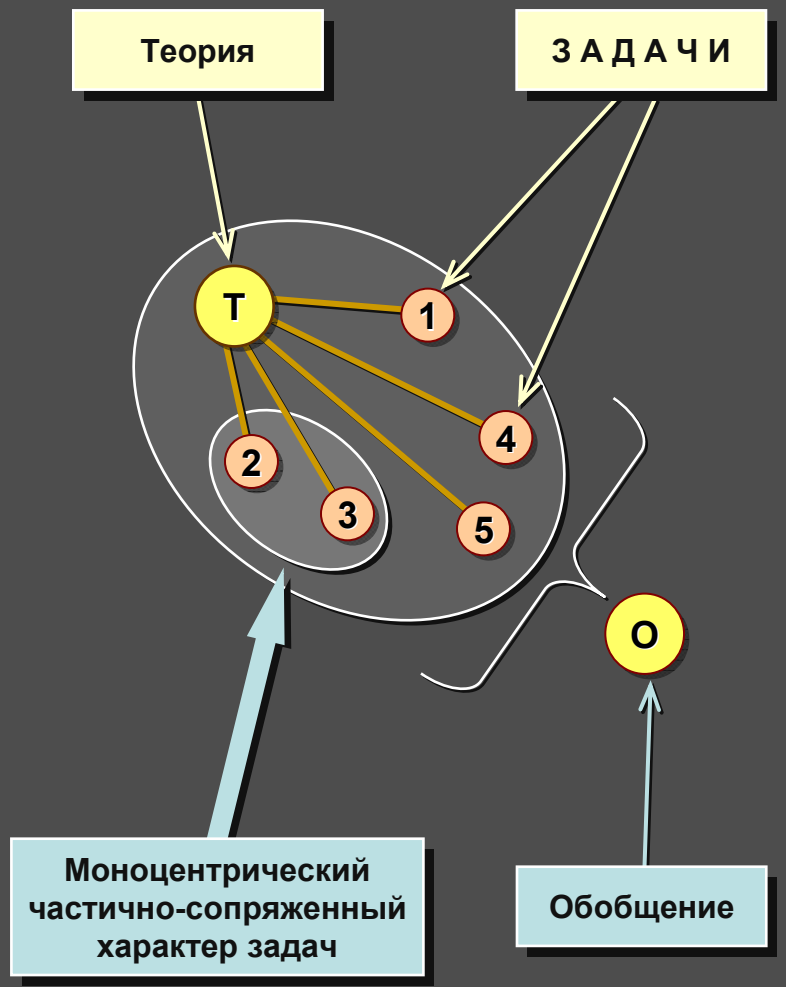


Олимпиадный путь Идеализированный (не реализуемый) путь Проектный путь

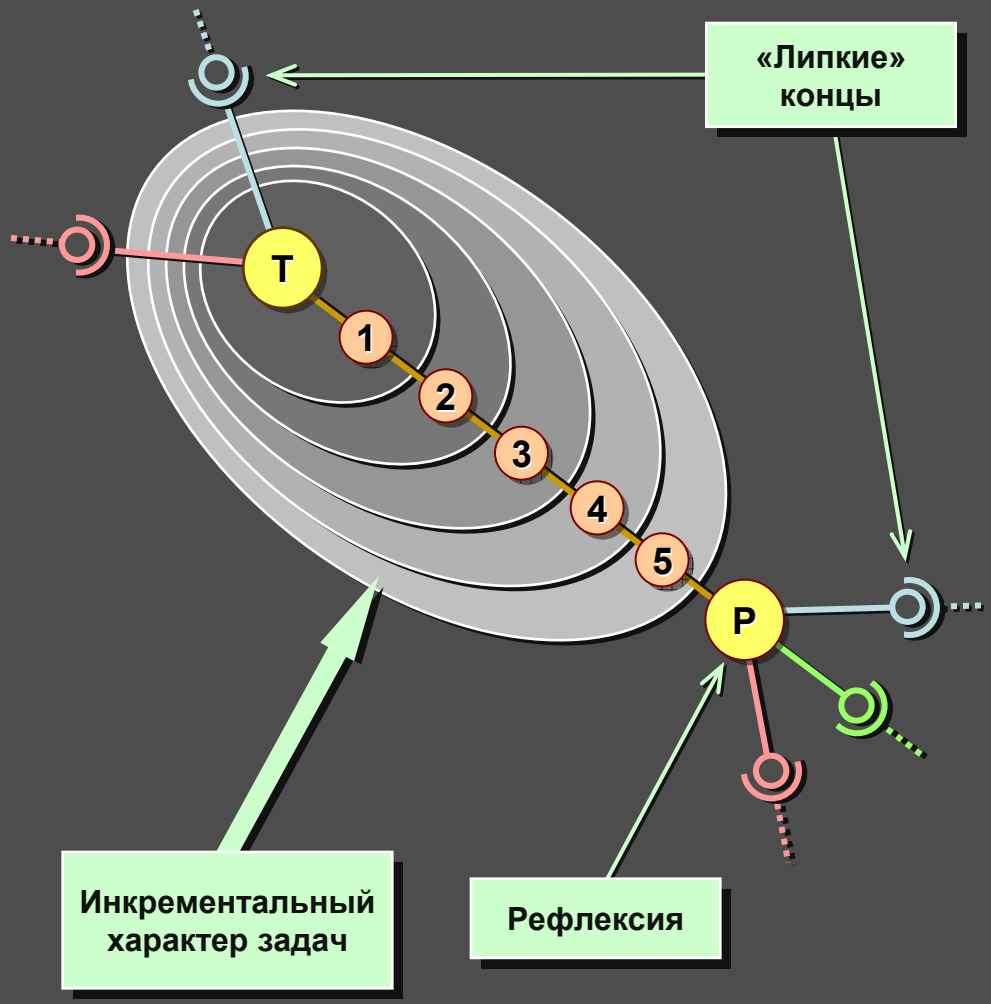
Воспользоваться учебной темой, как поводом для создания модельной ситуации

МИКРО-ПРОЕКТИРОВАНИЕ

ТРАДИЦИОННАЯ МЕТОДИКА

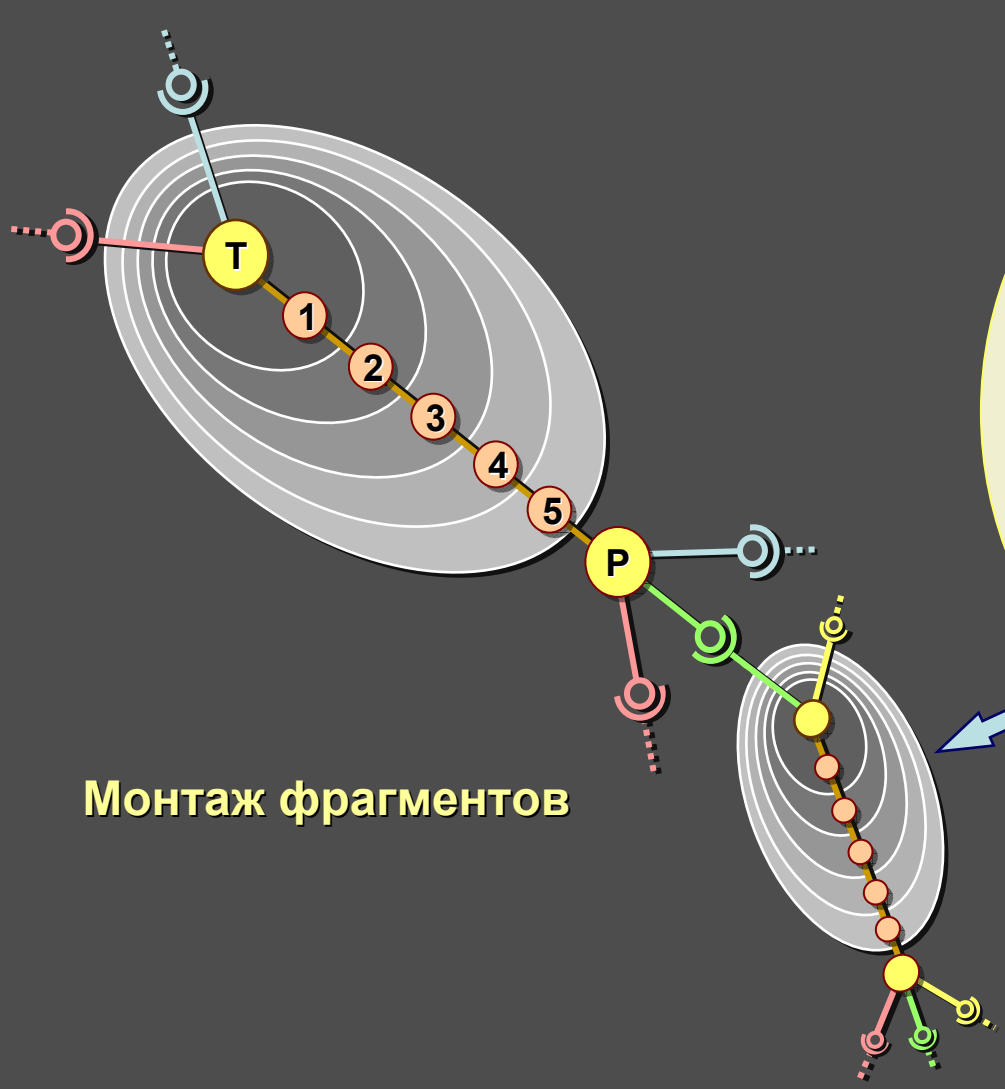


ПРОЕКТНАЯ МЕТОДИКА

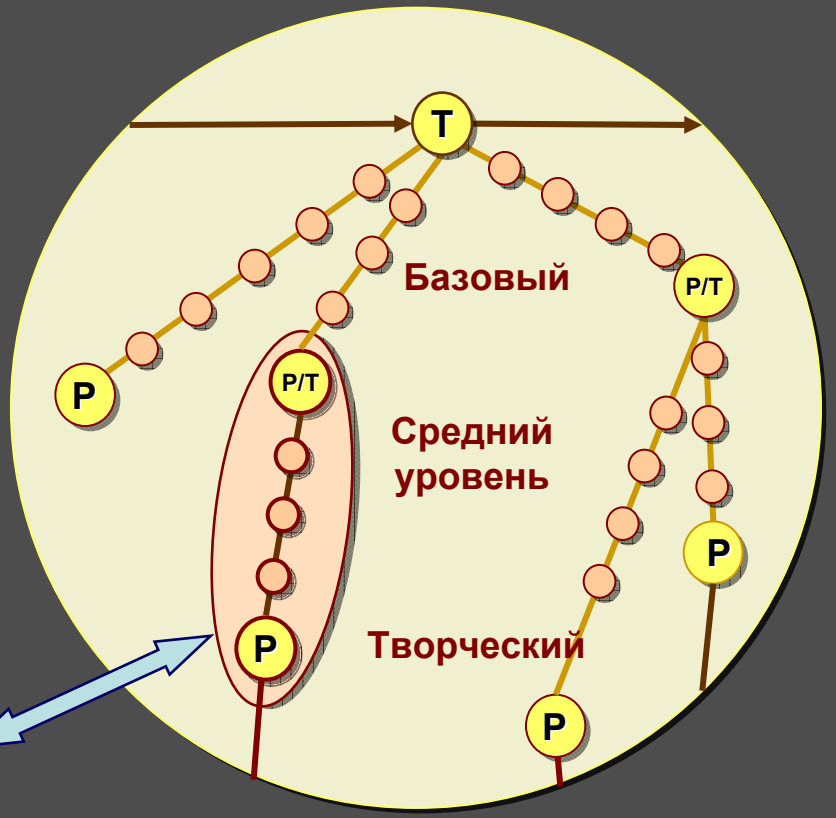


Монтажная единица – микропоследовательность («липкий фрагмент»)

МИКРО-ПРОЕКТИРОВАНИЕ



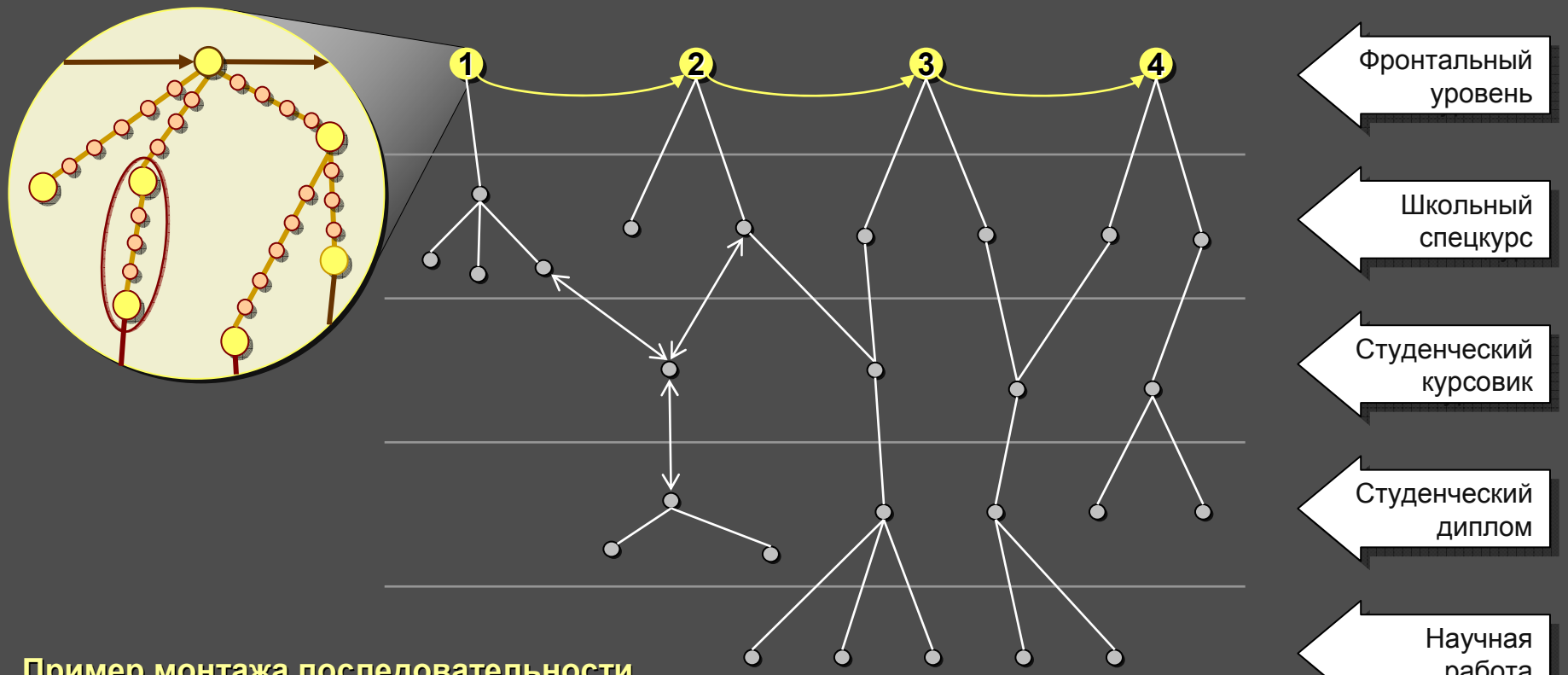
Монтаж фрагментов



Структура темы

T – теория, P – рефлексия

ФРОНТАЛЬНАЯ ПРОЕКТНАЯ СИСТЕМА



Пример монтажа последовательности

(1) Мультфильм

стиль, операторы, функции

(3) Графики

указатели, структуры, указатели на функции, классы, гр.интерфейс

(4) Молекулы

указатели, классы, наследование, гр.интерфейс

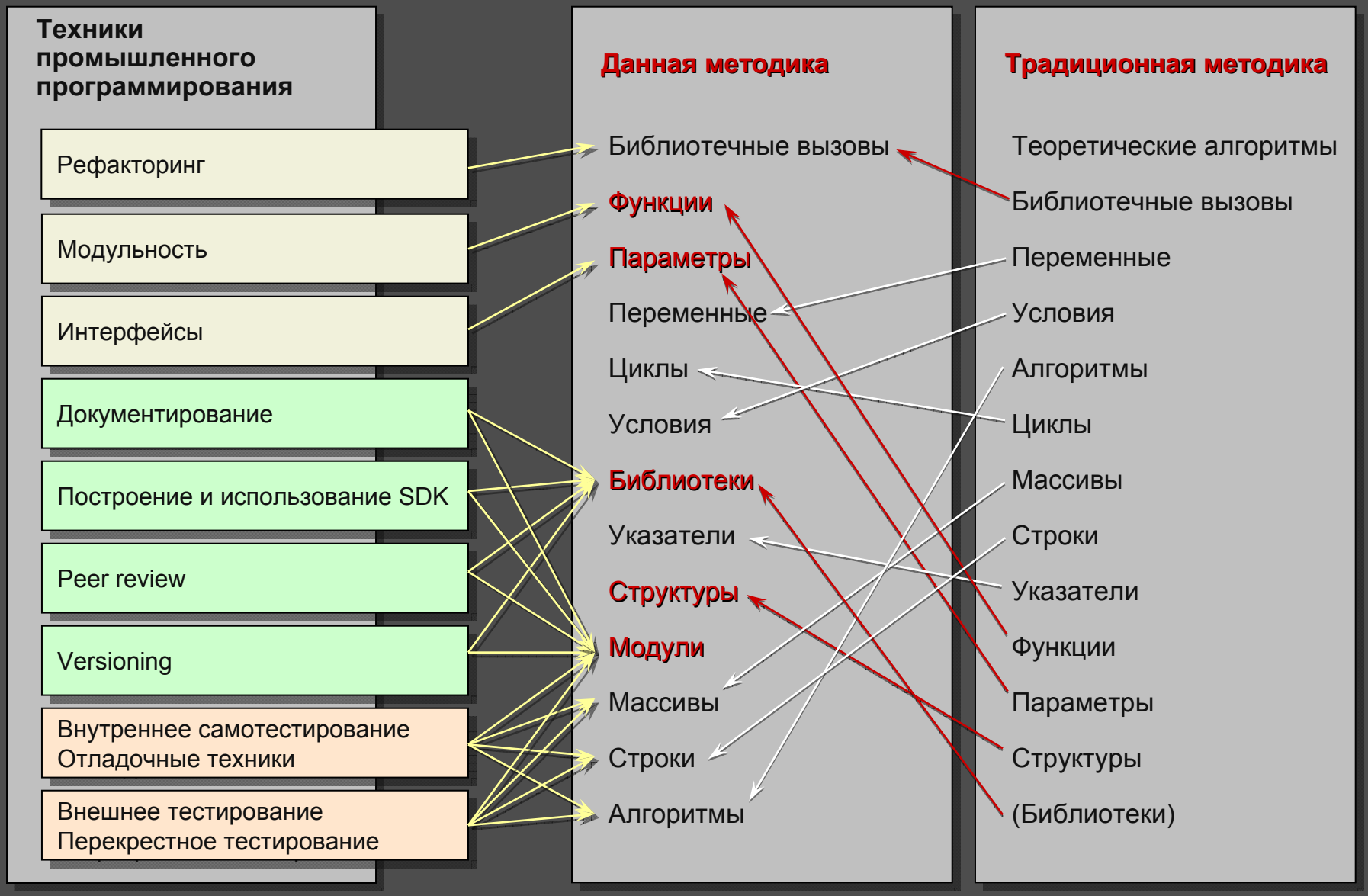
(2) Аркадные игры

указатели, структуры, указатели на функции, классы, наследование

(5) Оконная система

классы, наследование, проектирование, гр.интерфейс

ПОСЛЕДОВАТЕЛЬНОСТЬ ИЗУЧЕНИЯ



СТРУКТУРА КУРСА

7 класс
Основные понятия программирования
Базовая алгоритмика

- «Тупой художник»
- «Графики»
- «Возня с данными»
- Анализ сортировок
- Строковая библиотека

8 класс
Основы ООП, основы ООД, компонентное программирование, архитектурные навыки

- «ПРОЕКТ-МОЛЕКУЛА»
- Оконная система
- Компонентное программирование, плагины

9 класс
Алгоритмы и структуры данных, конструирование компиляторов. Язык ассемблера.

- Стек и стековая машина, наноассемблер
- Очереди, списки, хеш-таблицы
- Деревья
  - Экспертные системы
  - Архиватор (алг. Хаффмана)
  - Деревья выражений, nanoGCC
  - Символьное дифференцирование
  - Оптимизация выражений и программ
- Лексический анализ, графы и автоматы
- Нейронные сети

10 класс
Программирование на платформах Win32 и .NET
3D-графика, OpenGL и DirectX

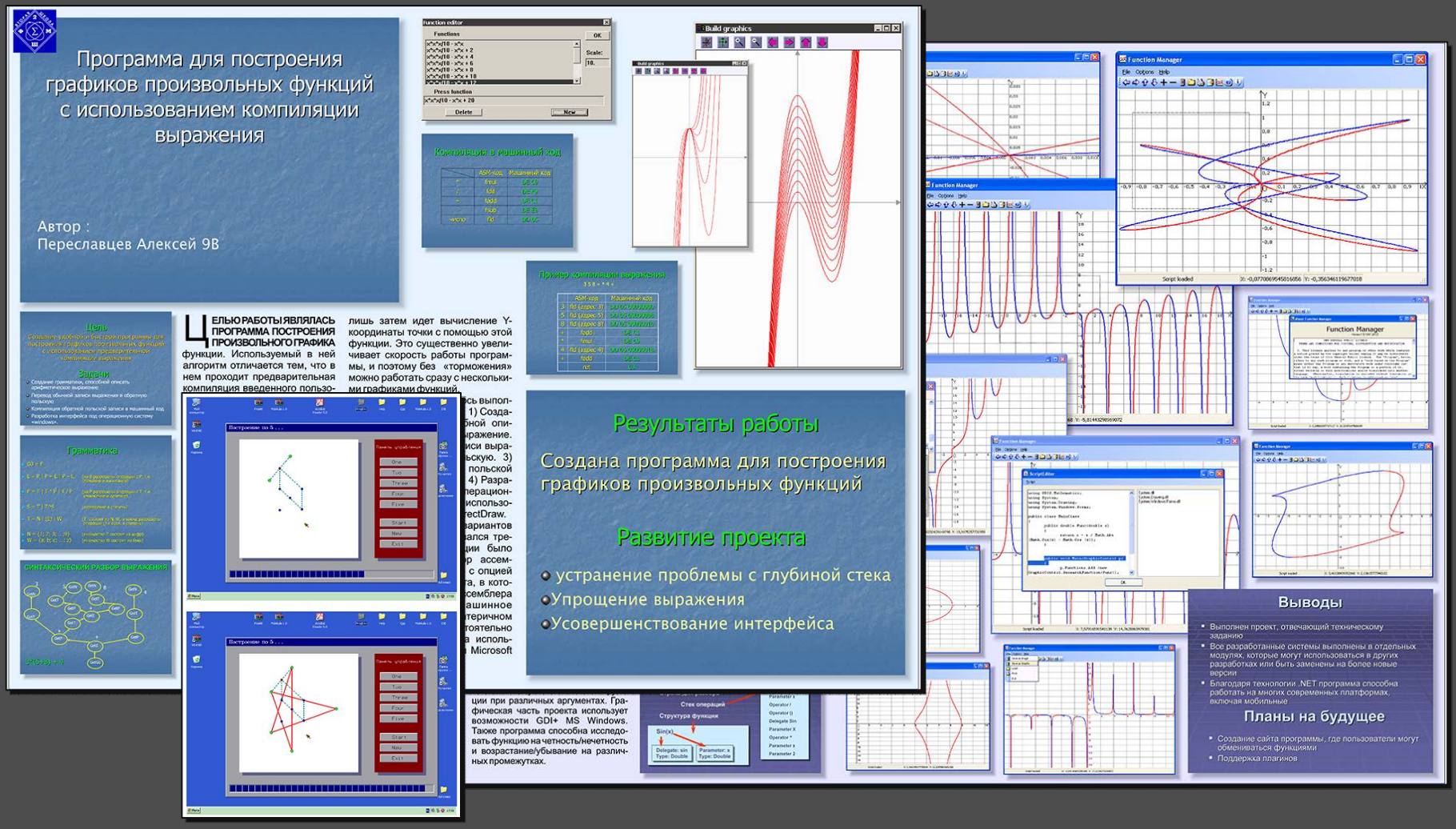
- Оконная система – оболочка для Win32
- Обработка данных, введение в численные методы, визуализация данных

11 класс
Проектная работа. Специальные пользовательские навыки. Борьба с ЕГЭ :(

2 ЧЕТВЕРТЬ – «ПОСТРОИТЕЛЬ ГРАФИКОВ»

Модуль пересчета координат + тонкая оболочка над графической библиотекой.

– Интерфейс («Кнопочки») – запланированная «головная боль». Набираем базу для MVC-архитектуры, классов и менеджеров интерфейса. Проект запросто углубляется до довольно серьезных вещей.



3-4 СЕМЕСТР – C++, МОДЕЛИРОВАНИЕ И ИНТЕРФЕЙСЫ

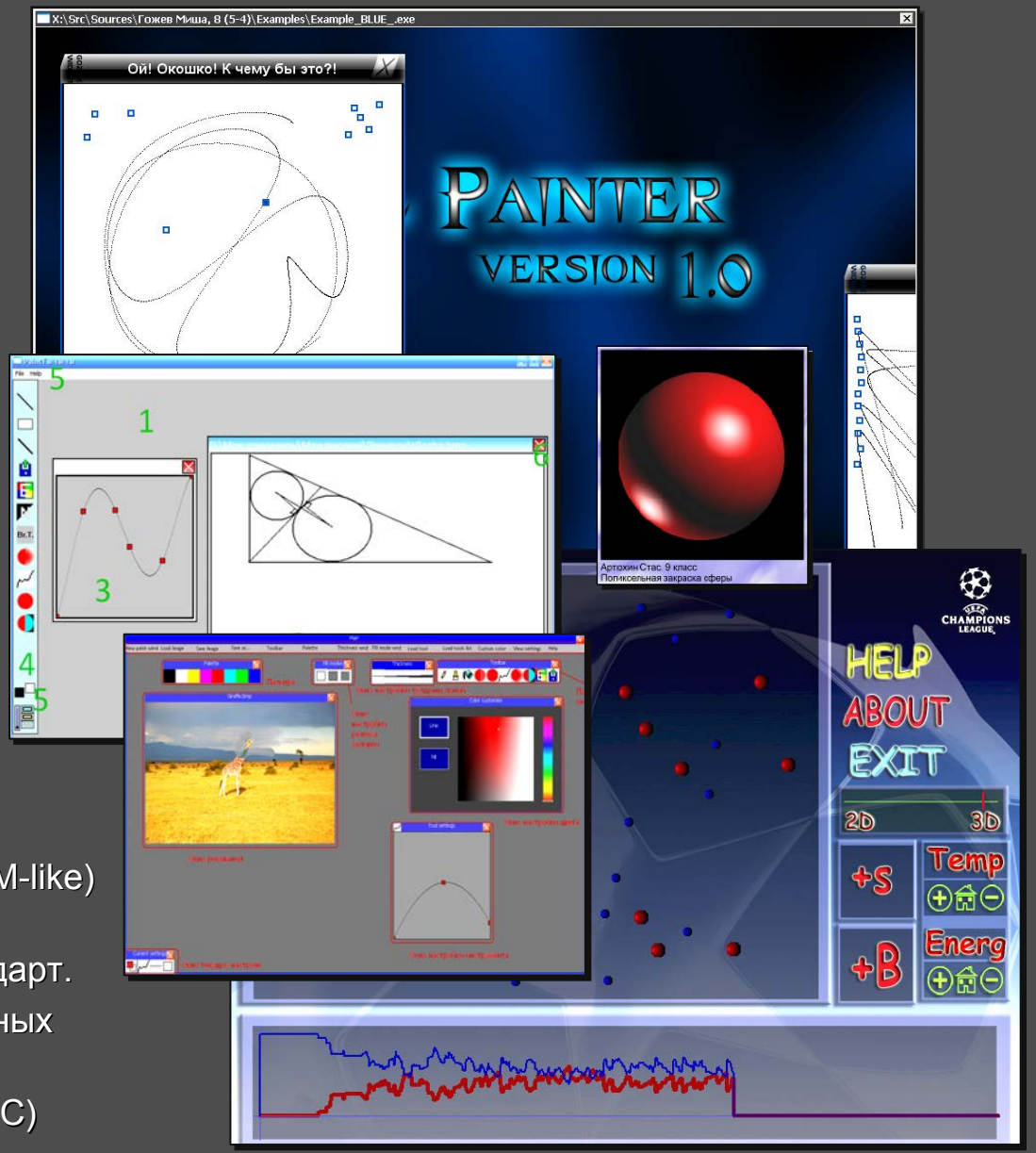
3 семестр – моделирование физики (идеальный газ, столкновения шаров, освещение) и химии (химическая кинетика).

4 семестр – классический проект GUI library – ООП «in large»

- ООД, затем ООП
- Start from UML, но не делая из него Holy Cow.
- Последовательное применение того, что учили в предыдущем семестре, но теперь не этюд, а проект.
- Невытесняющая многозадачность
- Многопоточность

Графический редактор nanoPhotoshop

- Плагины
- Компонентное программирование (COM-like)
- Единый стандарт API по группе, кто первый выпустил SDK – его и стандарт.
- Головная боль с поддержкой собственных старых версий. Еще +1 к рефлексии.
- Значительный объем кода (2-7 тыс. LOC)



5-6 СЕМЕСТР – СТРУКТУРЫ ДАННЫХ И АЛГОРИТМЫ

Моделирование вычислительных систем (ВС)
- Реализация простой виртуальной машины (VM):
  - CPU со стековой архитектурой, регистрами
  - У каждого – своя система команд
- Ассемблер и дизассемблер для своего CPU
- JIT-компиляция в Intel 8x86 (для продвинутых)
- Списковый процессор (nanoLISP)
- Особое внимание: надежность кода, диагностика

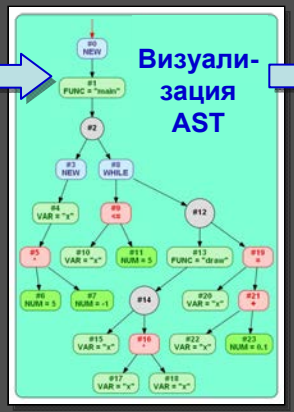
Проект nanoGCC
- Front-end для своего языка (нано-ЯВУ)
- Back-end для своего процессора
- Middle-end: простые оптимизации, символьное дифференцирование
- Трансляция $ЯВУ_1$ – $ЯВУ_2$ через AST

Групповая работа
- Стандарт AST, базовых операций

nanogcc.exe program.misha –m lena

misha_FE.exe	lena_BE.exe	lena_asm.exe	lena_vm.exe
program.misha →	tree.AST →	program.lena_asm →	program.lena → запуск на VM

```
function main
  {
  var x = -5;
  while (x <= 5)
    {
    draw (x, x*x);
    x = x + 0.1;
    }
  }
end
```
Скрипт Миши




```
.LENA
push -5
pop ax
:1
push ax
push +5
ja 2
push ax
push ax
push ax
mul

draw
push ax
push 0.1
add
pop ax
jmp 1
:2
end
```
Кросс-компиляция в Asm Лены

```
LENA V1.1
01 -5 12 01
11 01 01 +5
06 27 11 01
11 01 11 01
07 03 11 01
01 .1 04 12
01 05 04 00
```
Код CPU Лены

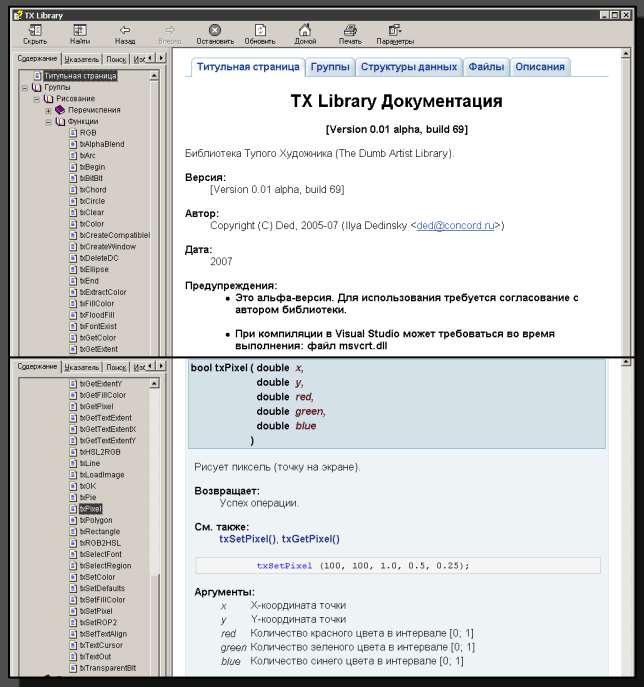
Структуры данных: Стек, Список, Хеш-Таблица, Дерево. Язык реализации – C++ или C.

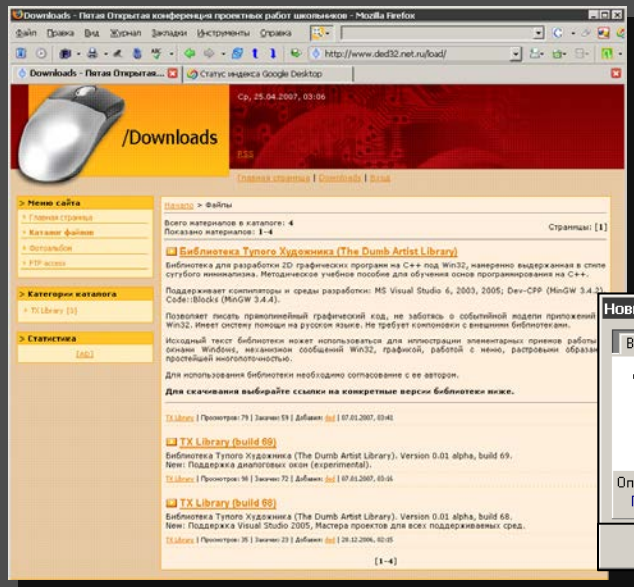
ВИЗУАЛЬНЫЙ ПОДХОД

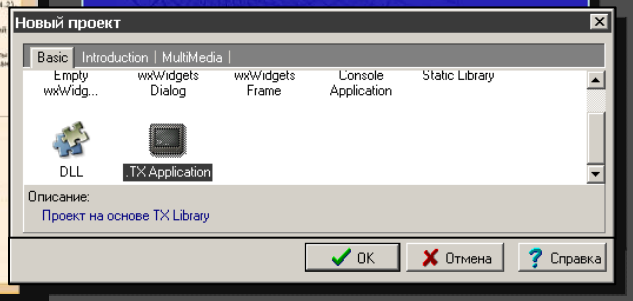
Максимизация мотивации, ясность результата

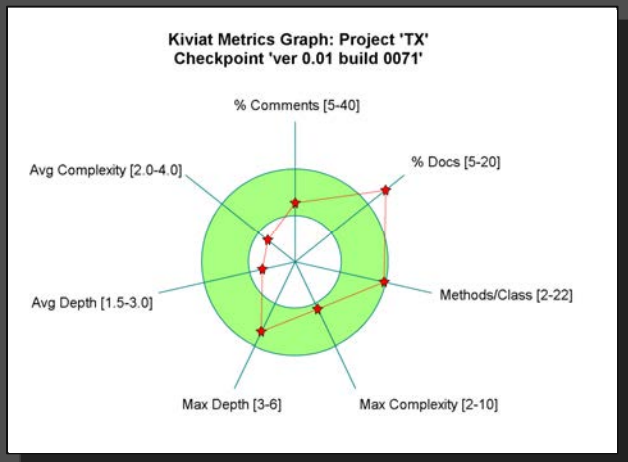
Разработана библиотека под Windows для облегчения работы

- Тонкая и прозрачная среда – реально разобраться
- Минимализм. Специально убрано все, что «вкусно», но можно сделать самому
- Без пафоса («песочница», «тупой художник», ироничная псевдореклама)
- Подталкивает «выйти за стенки песочницы»
- Хороший методический инструмент – тот, что попался в нужное время, но который не привязывает к себе, а побуждает идти дальше







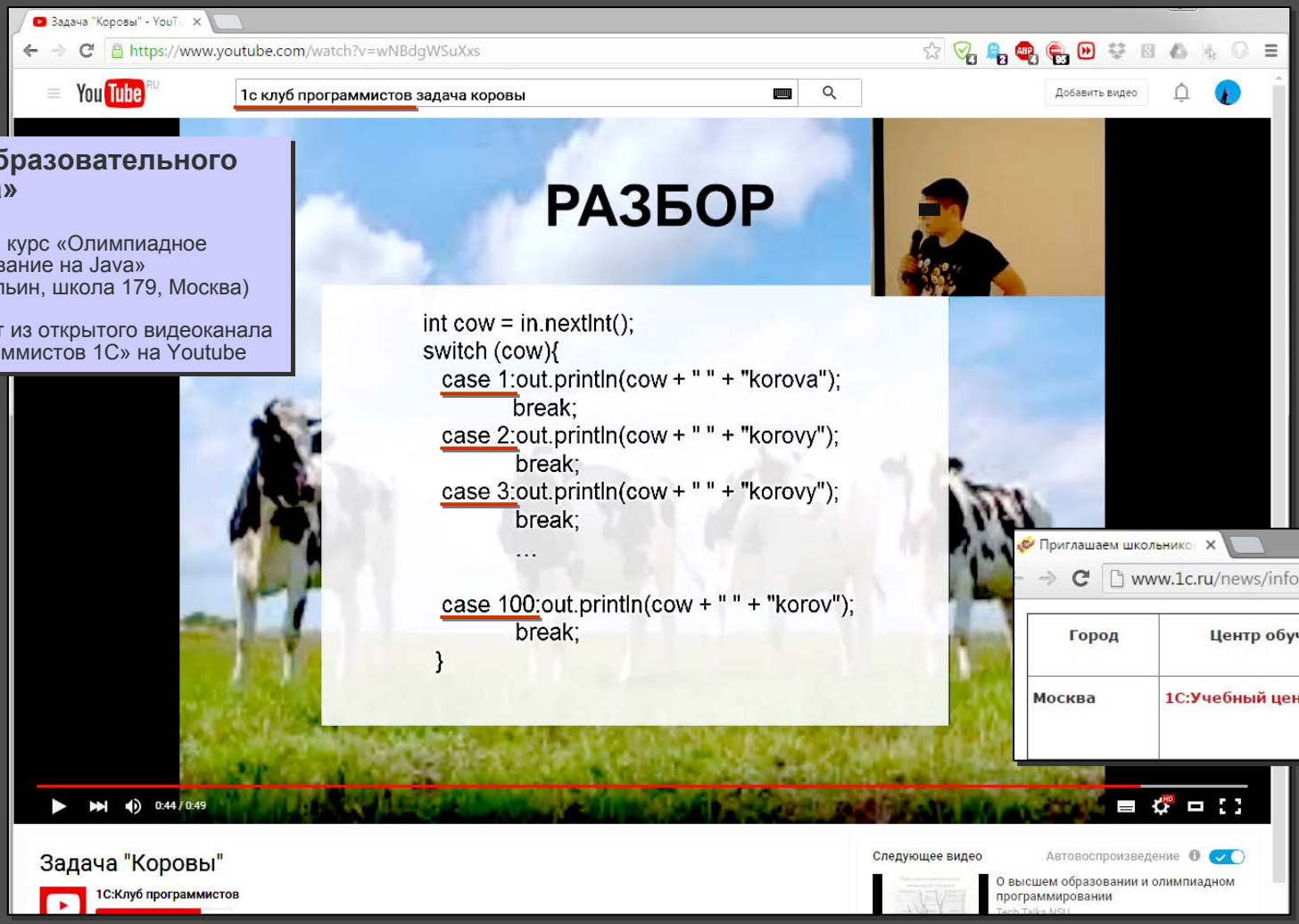


ПРОФЕССИОНАЛИЗМ ИЛИ ПРОФАНАЦИЯ*?

Пример образовательного «продукта»

(Компания 1С, курс «Олимпиадное программирование на Java» автор В. В. Ильин, школа 179, Москва)

Скриншот взят из открытого видеоканала «Клуба программистов 1С» на Youtube



В образовании легко навредить, ребенок ведь думает, что он успешен. В этом случае лучше – без программирования.

Часто господствует остаточный принцип: Для школы – «сойдет». А родители за это еще и заплатят.

* Оценочное суждение

О СОСТАВЛЯЮЩИХ КАЧЕСТВА

Проекта (почему он часто плох)?

1. Планирование	90%	90%
2. Алгоритмическая часть	90%	120%
3. Интерфейс	90%	100%
4. Система помощи	90%	80%
5. Веб-сайт	90%	80%
6. Презентация	90%	120%
7. Доклад	90%	120%
Результат = $\Pi\, r_i$	48%	100%

…и учебного процесса

Школьные приоритеты, позиция администрации	*a%*
Учебная нагрузка и расписание	*n%*
Мотивация детей и родителей	*m%*
Выстроенность системы курсов и требований	*s%*
Характеристики компьютеров и ПО	*h%*
Удобство пользования (настройки ПО и сети)	*u%*
Доска, проектор, стенды, знаменательные вещи	*v%*
Чистота и аккуратность в кабинете	*c%*
Личность учителя	*r %*
Результат	?

**Можно позволить много чего,
кроме убогого результата.
В этом случае лучше – без программирования.**

**Часто господствует принцип ДШС
(Для Школы – Сойдет).**

КОГДА НАЧИНАТЬ? СИСТЕМА НАБОРА

Наиболее критичный этап – старт обучения

- Сильно мотивационно обусловлен!
- Не только и не столько учить, сколько воспитывать

Требуемые качества

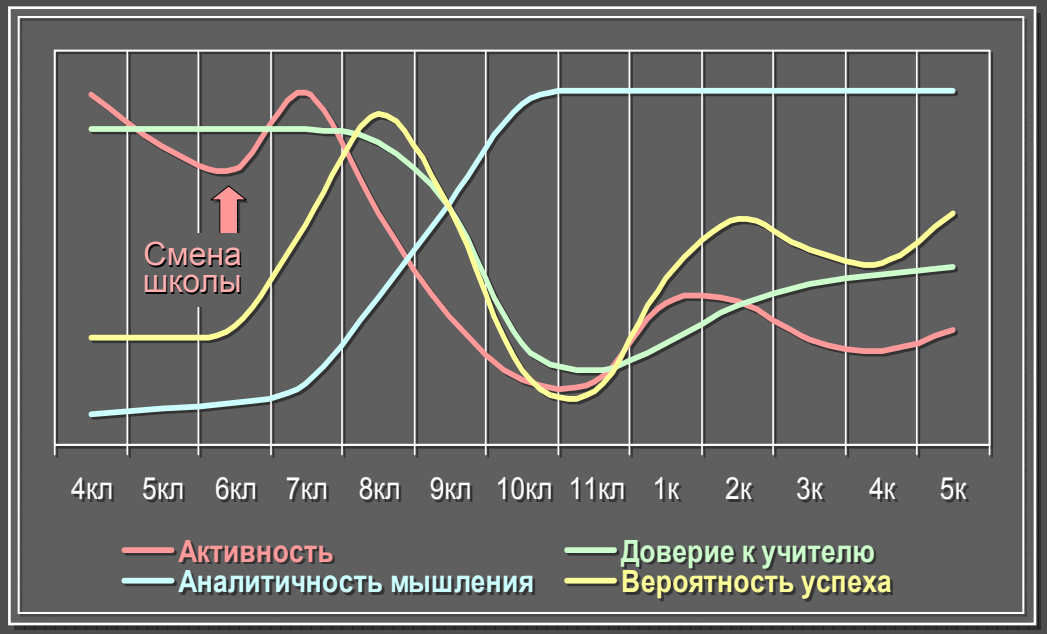
- Воображение, творчество
- Трудолюбие
- Обязательность
- Самооценка
- Результативность
- Коммуникабельность

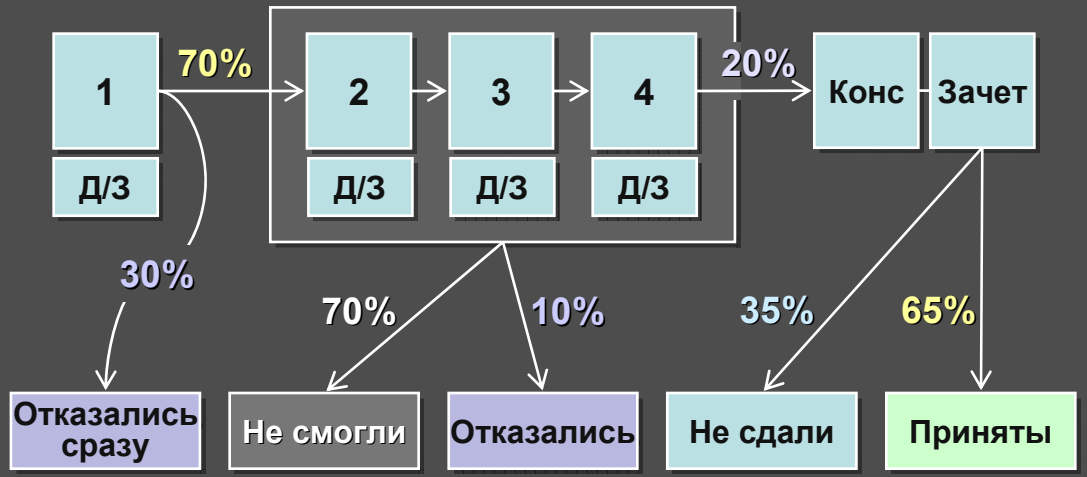
Система набора: микропроект (мультик)

- Лучше в 8 класс (сейчас в 7-й)
- Цель – попробовать процесс
- В любом случае – получение базовых знаний по программированию

Статистика набора

- 70% желали продолжать занятия
- из них 70% не смогли посетить занятия (из-за расписания вступительных экзаменов)





ОБРАЗОВАТЕЛЬНЫЕ ТРАЕКТОРИИ

Учебный процесс

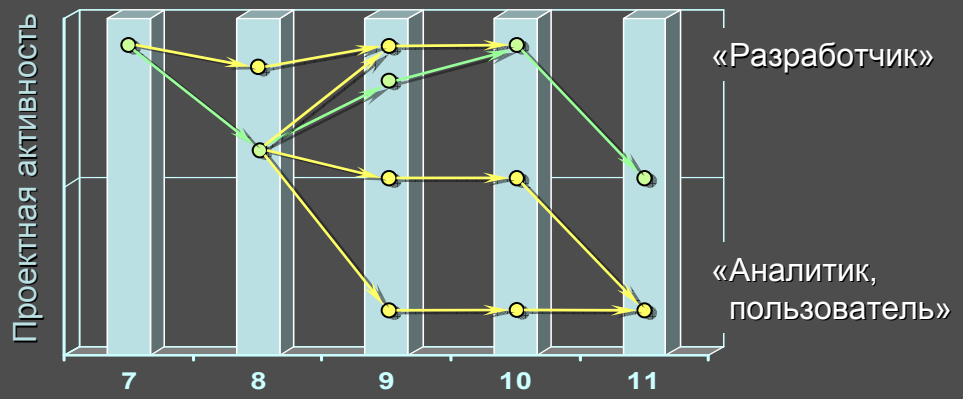
- Цельность
- **Непрерывность**
- Самовоспроизводимость
- Методическая база
- Техническая база

Ученик

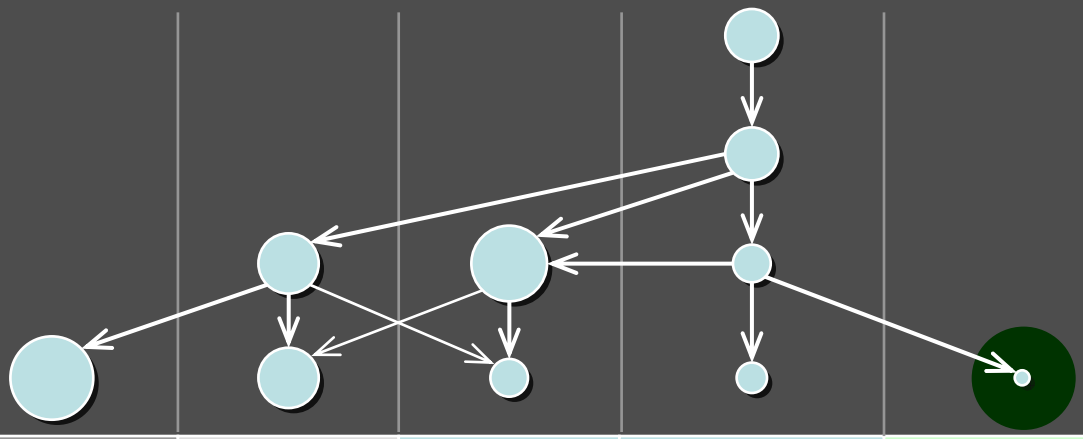
- Результативность
- Коммуникабельность
- **Самостоятельность**

Успех работы

Временная динамика



Где-то будет пауза… обычно на 2 году



Имитационная активность	Формальная активность	Олимпиадная активность	Проектная активность	Профессиональная активность
Лодырь	Ботан	Спортсмен	Экспериментатор, инженер	Профессионал

Непродуктивная активность

Продуктивная активность

1. МОТИВАЦИЯ

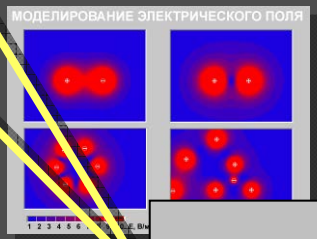
2. МЕТОДИКА

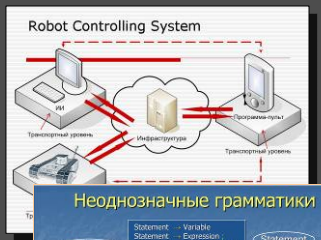
3. РЕЗУЛЬТАТЫ

ИТОГОВЫЕ РАБОТЫ

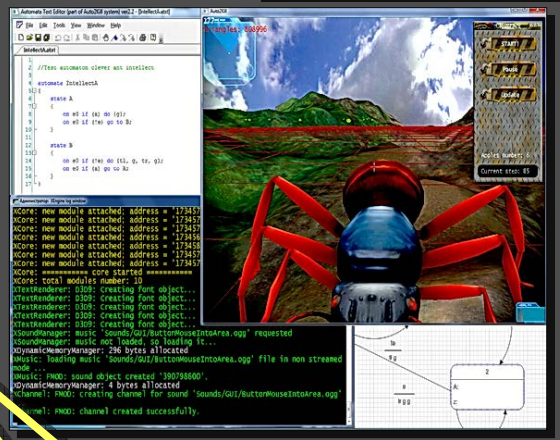
Математика

Физика



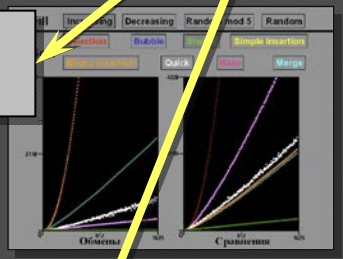


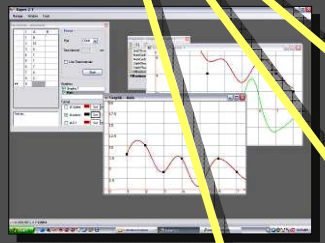
Технология



Проектное Программирование

География





Филология

Литература

Химия

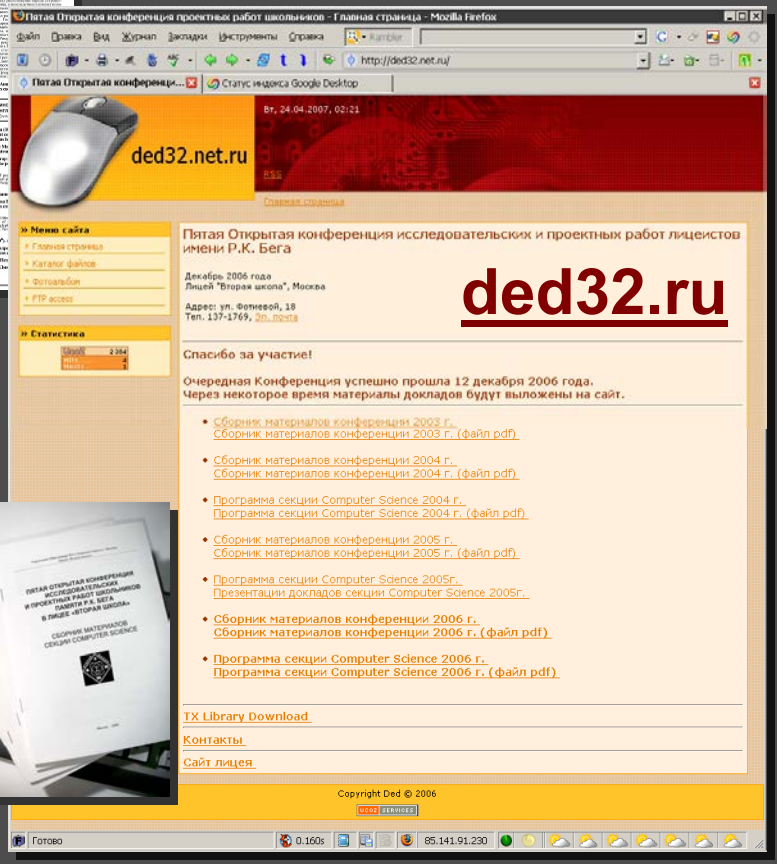
Биология

На слайде приведены фрагменты презентаций учащихся. См. ded32.ru

КОНФЕРЕНЦИЯ ПРОЕКТНЫХ РАБОТ


ТРЕТЬЯ ОТКРЫТАЯ НАУЧНО-ПРАКТИЧЕСКАЯ КОНФЕРЕНЦИЯ ЛИЦЕЯ

Конференция начиналась в 2003 г. с секций программирования и физики. Сейчас ее нет.

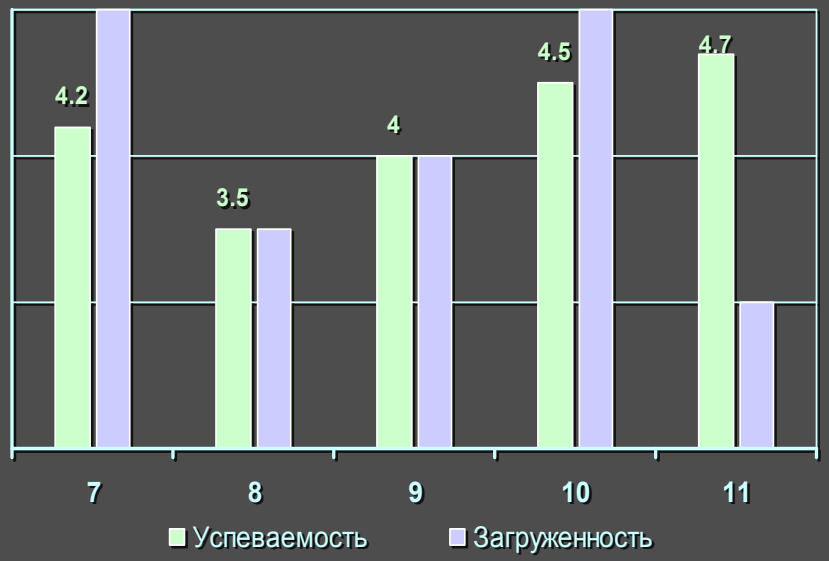


РЕЗУЛЬТАТЫ

Участие в региональных и международных конкурсах, статистика за 4 года (не считая призовых мест на олимпиадах регионального и Всероссийского уровня)

- **Балтийский научно-инженерный конкурс (Санкт-Петербург)**
  19 участников, 13 наград, 7-11 класс
- **Intel ISEF (Международный этап, США)**
  4 участника, 3 награды, 11 класс
- **Intel ISEF (Всероссийский этап, Москва)**
  9 участниов, 9 наград, 8-11 класс
- **«Старт в науку» (МФТИ, Москва)**
  16 участников, 15 наград, 8-11 класс

Динамика успеваемости по классам



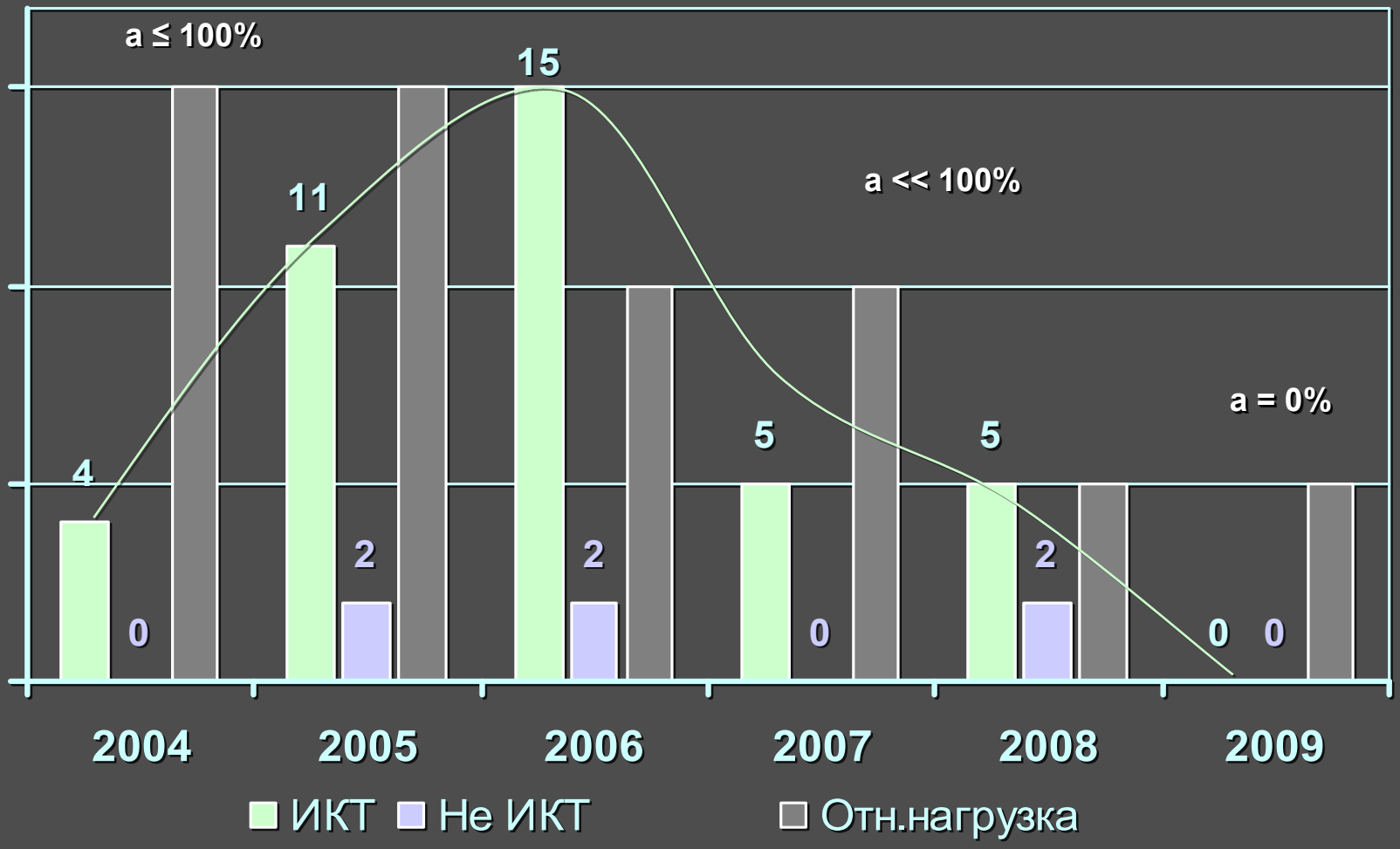
Сотрудничество с ВУЗами и НИИ РАН, учебными центрами компаний

- **МФТИ**
- **СПбГУ ИТМО**
- **ВМК МГУ**
- **Физфак МГУ**
- **Институт механики МГУ**
- **Институт Системного программирования РАН**
- **Компания «Компас-Плюс» (финансовое и банковское ПО)**
- **Intel**
- **Parallels**

В ходе обучения не использовались меры «нерезультативных – выгоним». Показатель ухода составлял в среднем 1-2 чел., прихода 0.5-1 чел. в год.

РЕЗУЛЬТАТЫ ОДНОГО ПРЕПОДАВАТЕЛЯ

Количество проектов на «Ярмарке Идей» (проектный конкурс ЮЗАО г.Москвы)



ЗАКЛЮЧЕНИЕ

В целом, методика показала свою эффективность и «нетравматичность».

Необходимые для нее условия:

- Методика работает в полную силу только при сильной математике и физике.
- Крайне желателен еще и английский язык.
- Методика работает в полную силу только при непрерывном поддержании мотивации от всех участников образовательного процесса.
- Она не может быть «слабым звеном», тогда она превратится в профанацию. Это опасно.
- «Гениальности» для успехов у ребенка не требуется. Но с ней будет легче получать результаты.
- Что требуется – работоспособность и коммуникабельность. К сожалению, эти качества трудно воспитываются в среднем и старшем школьном возрасте.
- Для успеха требуется свободное владение курсом физики и математики в объеме физматшколы.

Что дает методика

- Минимально – отсутствие проблем с информатикой/программированием в физико-математическом ВУЗе.
- Стандартно – умение программировать «для себя», с 1-2 курса ВУЗа успешно осваивать профильные занятия на 1-3 года вперед, часто опережая сокурсников.
- Программа по информатике МФТИ, ВМК и мехмата МГУ по программированию во многом покрыта стандартным курсом 7-10 классов.
- Максимально – посильной научной и наукоемкой производственной работой школьник может грамотно заняться начиная с 7-8 класса, с 10 класса быть приглашенным в учебно-научный или научно-производственный коллектив.
- Это «социальный лифт», позволяющий ребенку быстро подняться профессионально.

Спасибо за внимание

Больше информации на: http://ded32.ru

Благодарности

акад. В.П. Иванников (ИСП РАН)
член-корр. РАН А.Л. Семенов, проф. Н.И.Яковлева (МИОО)
проф. А.А. Шананин, проф. И.Б.Петров, доц. А.В. Гасников, Е.Г. Молчанов (МФТИ)
чл.-корр. РАО проф. В.Н. Васильев, проф. В.Г. Парфенов, проф. А.А. Шалыто и сотр. (СПбГУ ИТМО)
В.В. Никитин (Parallels), А.Л. Плоткин (Intel) А.С. Татаринов (nVidia)