

Карпаев Алексей, м.н.с. лаборатории физиологии человека

Современные технологии объектно-ориентированного программирования численных методов

Лекция 1

Обзор

```

procedure ALFA (KX, KY, Q, eps, T, P, dx, dy, J, K, M, nmax);
procedure KX, KY, Q, P; label M;
real eps, dx, dy;
integer J, K, nmax; array T;
Blok1:
begin
real alfa, kx, ky, almax, xy, yx, BO, DO, qO, C, G, R, max;
integer i, j, k, n, Kn, Kk;
array al [0:8], b, B [1:K, 0:J], f, H [0:K - 1, 0:J],
c, D [0:K, 1:J], e, F [0:K, 0:J - 1], E, q, d, V, dT [0:K, 0:J];
procedure min;
begin real a, a1;
a := 2/((1 + ky * xy / kx) * J * K);
a1 := 2 / ((1 + kx * yx / ky) * J * K);
min := if a < a1 then a else a1 end max;
Bibop alfa:
xy := dx/dy; yx := dy/dx;
begin almax := 0;
for k := 0 step 1 until K do
for j := 0 step 1 until J do
begin kx := KX(j * dx, k * dy);
ky := KY(j * dx, k * dy);
almax := almax + min end j, k;
almax := almax / ((J + 1) * (K + 1)) end NE 5;
for j := 0, 3, 6 do for i := 0, 1, 2 do al[j + i] := -almax * ((8 - j / 3 - 3 * i) / 8);
for k := 0 step 1 until K do
for j := 0 step 1 until J do begin if k = 0 then
B[k, j] := -KY(i * dx, (k - 1/2) * dy) * xy;
if j = 0 then D[k, j] := -KX((j - 1/2) * dx, k * dy);
xy; if k = K then H[k, j] := -KY(i * dx, (k + 1/2) * dy) * xy;
if j = J then F[k, j] := -KX((j + 1/2) * dx, k * dy) * yx;
E[k, j] := -((if k = K then H[k, j] else B[k, j]) + (if k = 0
then B[k, j] else H[k, j]) + (if j = 0 then D[k, j] else
F[k, j]) + (if j = J then F[k, j] else D[k, j]));
end k;
for j := 0 step 1 until J do
begin B[K, j] = 2 * B[K, j];

```

ЧТО делает данный код?

Стили программирования

Название функции содержит недостаточно информации о её деятельности:

Code Written By A CS 101 Student

```
public int fibonacci(int x) {
    if (x == 1) {
        return 1;
    } else if (x == 2) {
        return 1;
    } else {
        return fibonacci(x - 1) + fibonacci(x - 2);
    }
}
```

Название сообщает исчерпывающую информацию о деятельности:

Code Written At A Large Company

```
/**
 * getFibonacciNumber is a method that, given some index n, returns the nth
 * Fibonacci number.
 * @param n The index of the Fibonacci number you wish to retrieve.
 * @return The nth Fibonacci number.
 */
public CustomInteger64 getFibonacciNumber(CustomInteger64 n) {
    FibonacciDataViewBuilder builder =
        FibonacciDataViewBuilderFactory.createFibonacciDataViewBuilder(
            new FibonacciDataViewBuilderParams(n, null, null, 0, null));
    if (builder == FibonacciDataViewBuilderConstants.ERROR_STATE) {
        throw new FibonacciDataViewBuilderException();
    }
    FibonacciDataView dataView = builder.GenerateFibonacciDataView(this);
    if (dataView == FibonacciDataViewConstants.ERROR_STATE) {
        throw new FibonacciDataViewGenerationException();
    }
    return dataView.accessNextFibonacciNumber(null, null, null);
}
```

Зачем заботиться о стиле?

Стиль «написал-посчитал-выбросил»

- Код понятен в ϵ -окрестности времени написания программы
- Программа перестает быть понятной через ~ 1 месяц после написания

Код, написанный с использованием принципов ООП и «хорошего» стиля программирования

- Более ясный
- Удобней для дальнейшего использования и будущего расширения функциональности

Цель курса

- Недостаток времени на рассмотрение технологий программирования в курсе вычислительной математики
- Показать преимущества стиля ООП перед традиционным процедурным в реализации численных методов:
 - Более легкая поддержка кода
 - Красивый код
 - Более быстрое получение результатов

Парадигмы программирования

- **Процедурное программирование:** разбиение программы на процедуры, использование ранее написанных процедур
- **Структурное программирование** как более развитая версия процедурного: тщательное проектирование структуры программы с целью создания полностью автономных в смысле создания и отладки процедур
- **Объектно-ориентированное программирование:** объединение переменных и методов, соответствующим определенным объектам (человек, машина, записная книжка)

Основы ООП: классы и объекты

Класс:

- Поля (Data) — переменные класса
- Методы (Methods) — функции для работы с полями

Объект — экземпляр класса

Пример

Объявление класса

```
class Human:
```

Data:

Age

Methods:

SetAge(age)

GetAge()

Использование

```
...  
Human Ivan;  
Ivan.SetAge(age1);
```

```
Human Alexander;  
Alexander.SetAge(age2);  
...
```

```
if (Ivan.GetAge() < Alexander.GetAge())  
    do smth;
```

Основные принципы ООП

- Абстракция
- Инкапсуляция
- Наследование
- Полиморфизм

Абстракция

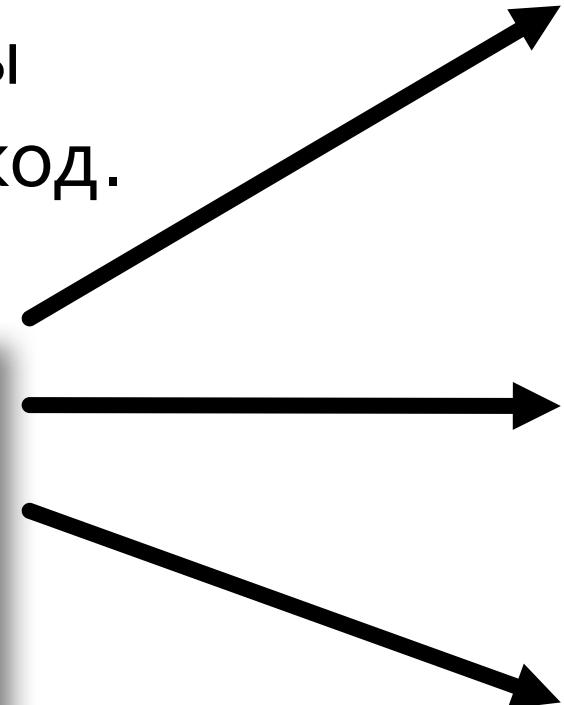
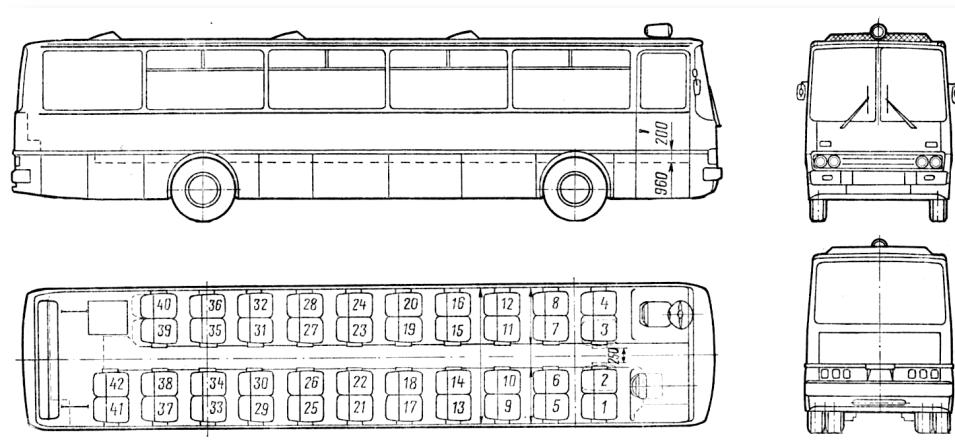
- Позволяет определить, какие характеристики (поля, методы) должен иметь объект, а какими из них можно пренебречь в контексте своего использования в конкретной программе
- **Пример:** человек как
 - работник в организации
 - студент в университете
 - гражданин в государстве

Инкапсуляция

- Принцип сокрытия данных от внешнего воздействия.
Обеспечивает безопасность пользования программой
сторонними пользователями
- Разделение прав доступа к полям класса на **private** и
public

Наследование

- Позволяет избежать повторения кода при расширении функциональности программы используя ранее написанный код.



Полиморфизм

- Способность объектов с одним названием иметь различную реализацию
- Примеры:
 - C++: *int add(int a, int b)* и *double add(double a, double b)*
 - Функция *MakeTimestep(AbstractSolver solver, Equation eq)* в решателе ОДУ: в качестве первого аргумента можно передать объект-наследник от класса *AbstractSolver*, например *ExplicitEulerSolver*, *RungeKuttaSolver*, *AdamsSolver*, ...

ИТОГ

- Познакомились с ООП
- Основные принципы ООП: **абстракция, инкапсуляция, наследование и полиморфизм**
- В курсе будет использоваться язык



Темы, которые будут рассмотрены

- Введение в Python
- Тренировочные задачи по ООП
- ООП реализации стандартных задач вычислительной математики: **численное дифференцирование, интегрирование, решение ОДУ и уравнений в ЧП**
- Обзор структуры крупных программных комплексов для решения вычислительных задач
- Использование средств ускорения Python

Темы, которые будут рассмотрены

- Введение в Python
- Тренировочные задачи по ООП
- ООП реализации стандартных задач вычислительной математики: **численное дифференцирование, интегрирование, решение ОДУ и уравнений в ЧП**
- Обзор структуры крупных программных комплексов для решения вычислительных задач
- Использование средств ускорения Python