Лекция 9
Построение абстракций и декомпозиция

Пименов Евгений Сергеевич

Курс «Программирование»

Сибирский государственный университет телекоммуникаций и информатики (Новосибирск)

Осенний семестр, 2016

О сложности

> *There is no single development, in either technology or management technique, which by itself promises even one order of magnitude [tenfold] improvement within a decade in productivity, in reliability, in simplicity*
>
> Frederick Phillips Brooks, «No silver bullet»

О сложности

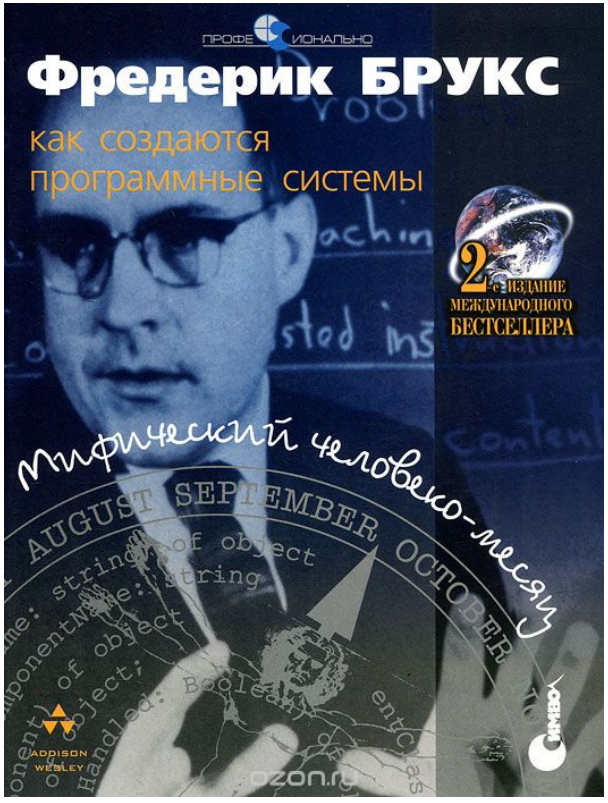
> *Нет ни одного открытия ни в технологии, ни в методах управления, одно только использование которого обещало бы в течение ближайшего десятилетия на порядок повысить производительность, надежность, простоту разработки программного обеспечения.*
>
> Frederick Phillips Brooks, «No silver bullet»

О сложности

- accidental complexity — сложность, привносимая разработчиком
- essential complexity — сложность решаемой задачи

О сложности

> *Управление сложностью — квинтэссенция программирования*
>
> B. Kernighan



Фредерик Брукс

«Мифический человеко-месяц»

Абстракции

Абстракция — способ сокрытия деталей реализации определенного набора функциональных возможностей.

Абстракции

Абстракция — способ сокрытия деталей реализации определенного набора функциональных возможностей.

Примеры:

- printf

Абстракции

Абстракция — способ сокрытия деталей реализации определенного набора функциональных возможностей.

Примеры:

- printf
- Ввод-вывод в Unix

Абстракции

Абстракция — способ сокрытия деталей реализации определенного набора функциональных возможностей.

Примеры:

- printf
- Ввод-вывод в Unix
- Язык C

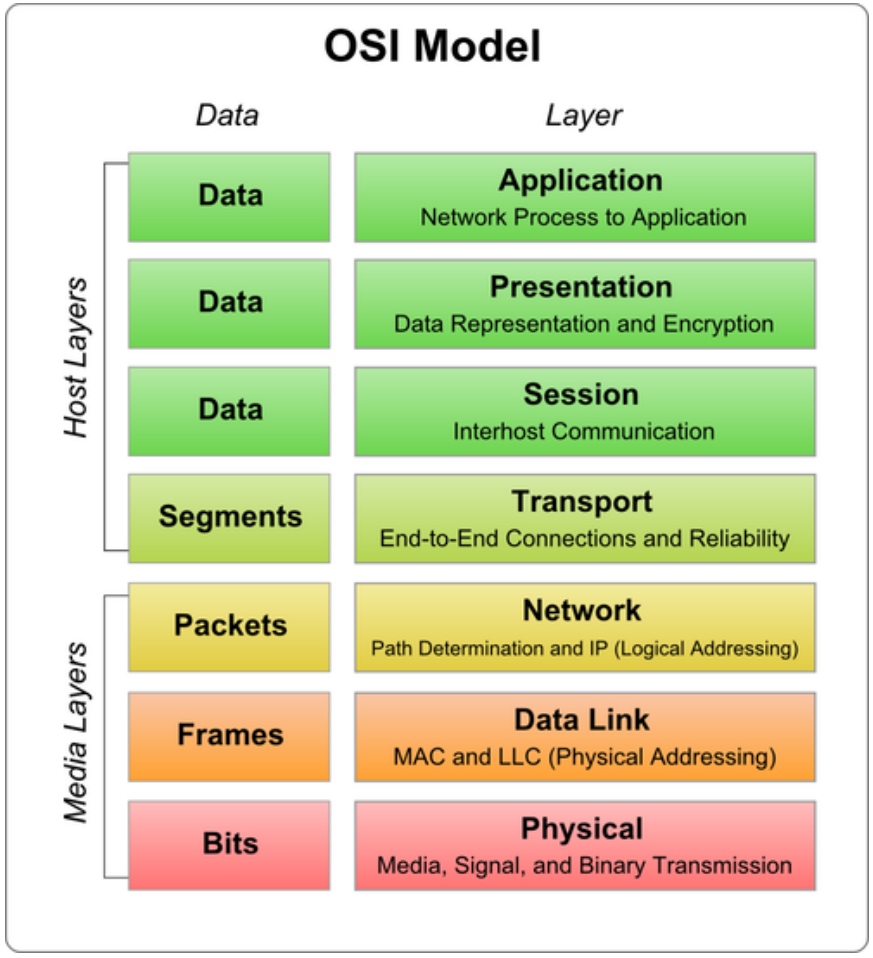
Абстракции

Абстракция — способ сокрытия деталей реализации определенного набора функциональных возможностей.

Примеры:

- printf
- Ввод-вывод в Unix
- Язык C
- Модель OSI
- ...

Модель OSI


OSI Model
Data
Layer
Host Layers
Media Layers
Data
Application
Network Process to Application
Data
Presentation
Data Representation and Encryption
Data
Session
Interhost Communication
Segments
Transport
End-to-End Connections and Reliability
Packets
Network
Path Determination and IP (Logical Addressing)
Frames
Data Link
MAC and LLC (Physical Addressing)
Bits
Physical
Media, Signal, and Binary Transmission

Абстракции

> *Все нетривиальные абстракции дырявы.*
>
> Джоел Спольски, Закон дырявых абстракций

http://russian.joelonsoftware.com/Articles/LeakyAbstractions.html

Абстракции

Некоторые инструменты для построения абстракций в языке C:

- Функции
- Макросы
- Структуры
- Синонимы типов

Best practice

- Одна задача — одна функция
- Использование чистых функций
- Разделение данных, логики и представления

Чистые функции

Чистая функция:

- Является детерминированной
- Не обладает побочными эффектами

Пример функции с побочным эффектом


```
int arg = 0;


int f(int n) {
    if (arg > 0) {
        arg = 0;
        return h(n);
    }
    arg = 1;
    return g(n);
}
```

Устранение побочного эффекта


```
int f(int n, int arg) {
    if (arg > 0) {
        arg = 0;
        return h(n);
    }
    arg = 1;
    return g(n);
}
```

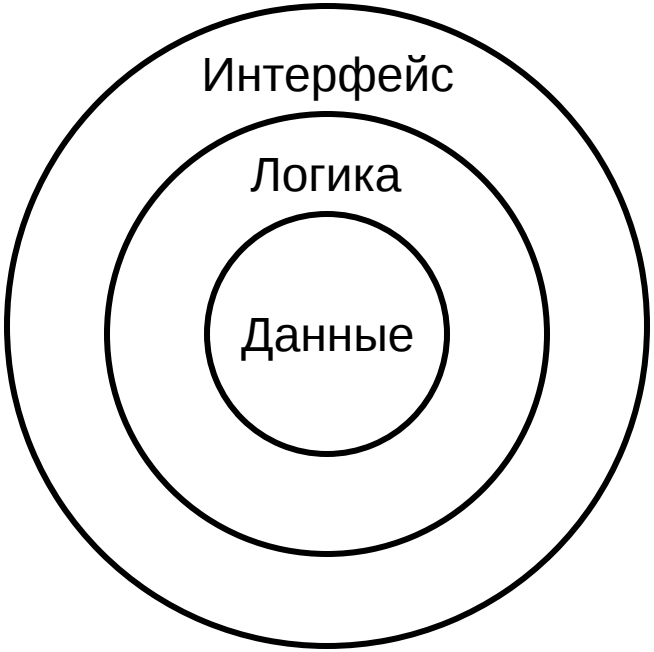
Разделение логики и представления


```
void sum(int a, int b) {
    printf("%d\n", a + b);
}
```

Разделение логики и представления


```
int sum(int a, int b) {
    return a + b;
}

...

printf("a + b = %d\n", sum(a, b));
MessageBox(sum(a, b));
SendRequest(sum(a, b));
```

Абстрактная модель программы



MV* patterns

- MVC — Model-View-Controller
- MVVM — Model-View-ViewModel
- MVP — Model-View-Presenter

Различия между MVVM и остальными MV*-паттернами

https://habrahabr.ru/company/mobileup/blog/313538/

SOLID

- The Single Responsibility Principle
- The Open Closed Principle
- The Liskov Substitution Principle
- The Interface Segregation Principle
- The Dependency Inversion Principle

Стив Макконнел

«Совершенный код»

Пример


```
bool f(
        float, float,
        float, float,
        float, float,
        float, float,
        float, float);
```

- Что делает эта функция?
- Как ее использовать?

Пример


```
bool f(
        float rx1, float ry1,
        float rx2, float ry2,
        float rx3, float ry3,
        float rx4, float ry4,
        float rx5, float ry5);
```

- Что делает эта функция?
- Как ее использовать?

Пример


```
bool f(
        float rx1, float ry1,
        float rx2, float ry2,
        float rx3, float ry3,
        float rx4, float ry4,
        float rx5, float ry5)
{
    bool t1_1 = ((rx5 - rx2) * (ry1 - ry2) - (rx1 - rx2) * (ry5 - ry2)) < 0;
    bool t1_2 = ((rx5 - rx3) * (ry2 - ry3) - (rx2 - rx3) * (ry5 - ry3)) < 0;
    bool t1_3 = ((rx5 - rx1) * (ry3 - ry1) - (rx3 - rx1) * (ry5 - ry1)) < 0;

    bool t2_1 = ((rx5 - rx3) * (ry1 - ry3) - (rx1 - rx3) * (ry5 - ry3)) < 0;
    bool t2_2 = ((rx5 - rx4) * (ry3 - ry4) - (rx3 - rx4) * (ry5 - ry4)) < 0;
    bool t2_3 = ((rx5 - rx1) * (ry4 - ry1) - (rx4 - rx1) * (ry5 - ry1)) < 0;

    return (t1_1 == t1_2 && t1_2 == t1_3) || (t2_1 == t2_2 && t2_2 == t2_3);
}
```

Пример


```
bool is_point_in_rectangle(
        float rx1, float ry1,
        float rx2, float ry2,
        float rx3, float ry3,
        float rx4, float ry4,
        float rx5, float ry5)
{
    bool t1_1 = ((rx5 - rx2) * (ry1 - ry2) - (rx1 - rx2) * (ry5 - ry2)) < 0;
    bool t1_2 = ((rx5 - rx3) * (ry2 - ry3) - (rx2 - rx3) * (ry5 - ry3)) < 0;
    bool t1_3 = ((rx5 - rx1) * (ry3 - ry1) - (rx3 - rx1) * (ry5 - ry1)) < 0;

    bool t2_1 = ((rx5 - rx3) * (ry1 - ry3) - (rx1 - rx3) * (ry5 - ry3)) < 0;
    bool t2_2 = ((rx5 - rx4) * (ry3 - ry4) - (rx3 - rx4) * (ry5 - ry4)) < 0;
    bool t2_3 = ((rx5 - rx1) * (ry4 - ry1) - (rx4 - rx1) * (ry5 - ry1)) < 0;

    return (t1_1 == t1_2 && t1_2 == t1_3) || (t2_1 == t2_2 && t2_2 == t2_3);
}
```

Пример


```
bool is_point_in_rectangle(
        float rx1, float ry1,
        float rx2, float ry2,
        float rx3, float ry3,
        float rx4, float ry4,
        float rx5, float ry5)
{
    bool t1_1 = ((rx5 - rx2) * (ry1 - ry2) - (rx1 - rx2) * (ry5 - ry2)) < 0;
    bool t1_2 = ((rx5 - rx3) * (ry2 - ry3) - (rx2 - rx3) * (ry5 - ry3)) < 0;
    bool t1_3 = ((rx5 - rx1) * (ry3 - ry1) - (rx3 - rx1) * (ry5 - ry1)) < 0;

    bool t2_1 = ((rx5 - rx3) * (ry1 - ry3) - (rx1 - rx3) * (ry5 - ry3)) < 0;
    bool t2_2 = ((rx5 - rx4) * (ry3 - ry4) - (rx3 - rx4) * (ry5 - ry4)) < 0;
    bool t2_3 = ((rx5 - rx1) * (ry4 - ry1) - (rx4 - rx1) * (ry5 - ry1)) < 0;

    return (t1_1 == t1_2 && t1_2 == t1_3) || (t2_1 == t2_2 && t2_2 == t2_3);
}
```

Кажется, стало понятнее...

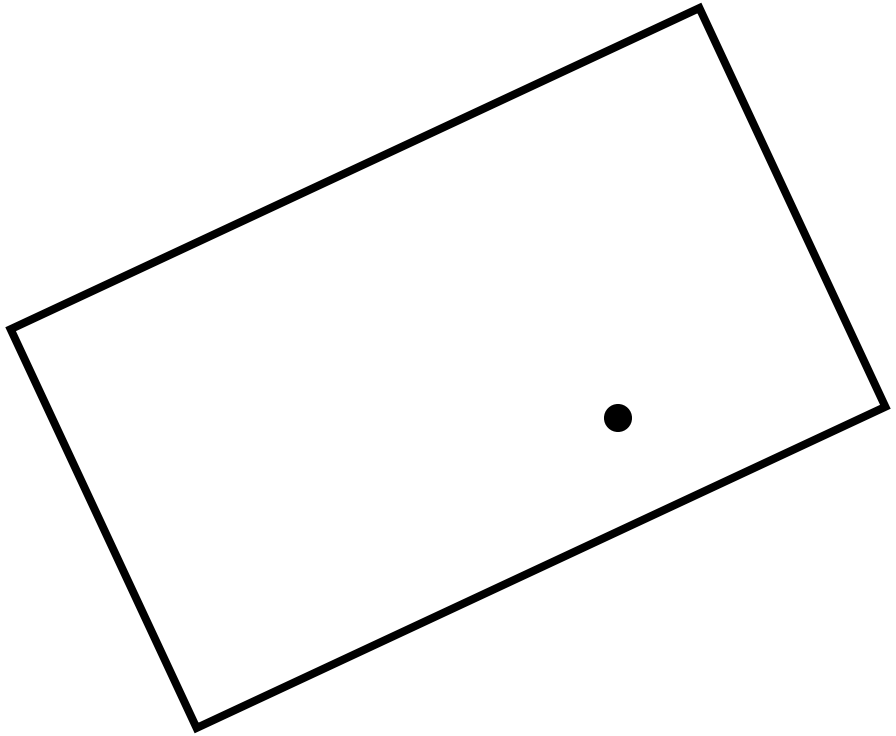
Пример


```
bool is_point_in_rectangle(
        float rx1, float ry1,
        float rx2, float ry2,
        float rx3, float ry3,
        float rx4, float ry4,
        float rx5, float ry5)
{
    bool t1_1 = ((rx5 - rx2) * (ry1 - ry2) - (rx1 - rx2) * (ry5 - ry2)) < 0;
    bool t1_2 = ((rx5 - rx3) * (ry2 - ry3) - (rx2 - rx3) * (ry5 - ry3)) < 0;
    bool t1_3 = ((rx5 - rx1) * (ry3 - ry1) - (rx3 - rx1) * (ry5 - ry1)) < 0;

    bool t2_1 = ((rx5 - rx3) * (ry1 - ry3) - (rx1 - rx3) * (ry5 - ry3)) < 0;
    bool t2_2 = ((rx5 - rx4) * (ry3 - ry4) - (rx3 - rx4) * (ry5 - ry4)) < 0;
    bool t2_3 = ((rx5 - rx1) * (ry4 - ry1) - (rx4 - rx1) * (ry5 - ry1)) < 0;

    return (t1_1 == t1_2 && t1_2 == t1_3) || (t2_1 == t2_2 && t2_2 == t2_3);
}
```

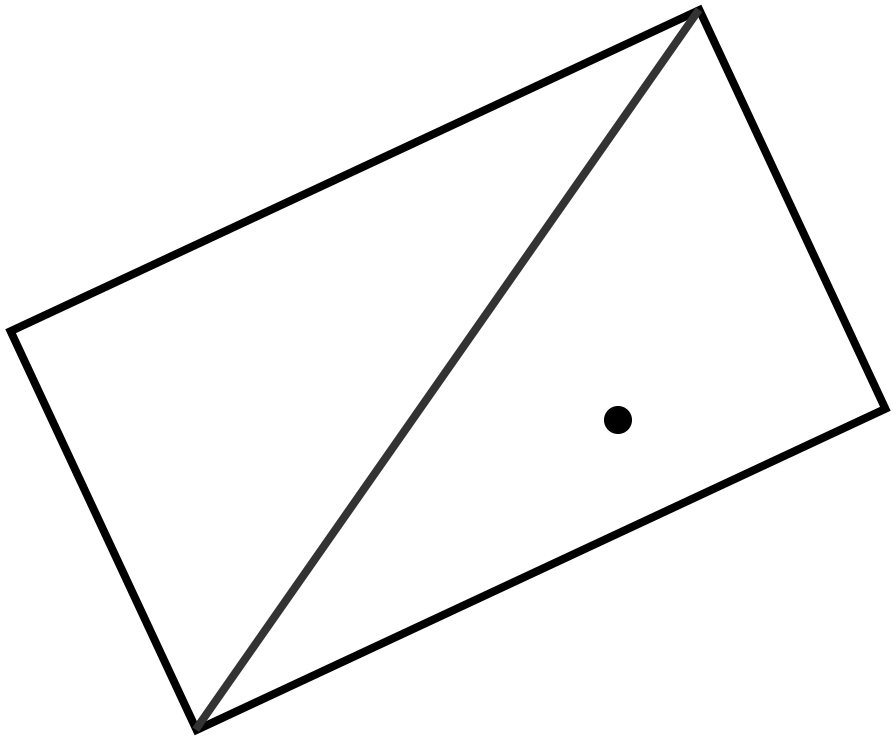
Кажется, стало понятнее...

Но как этим пользоваться?

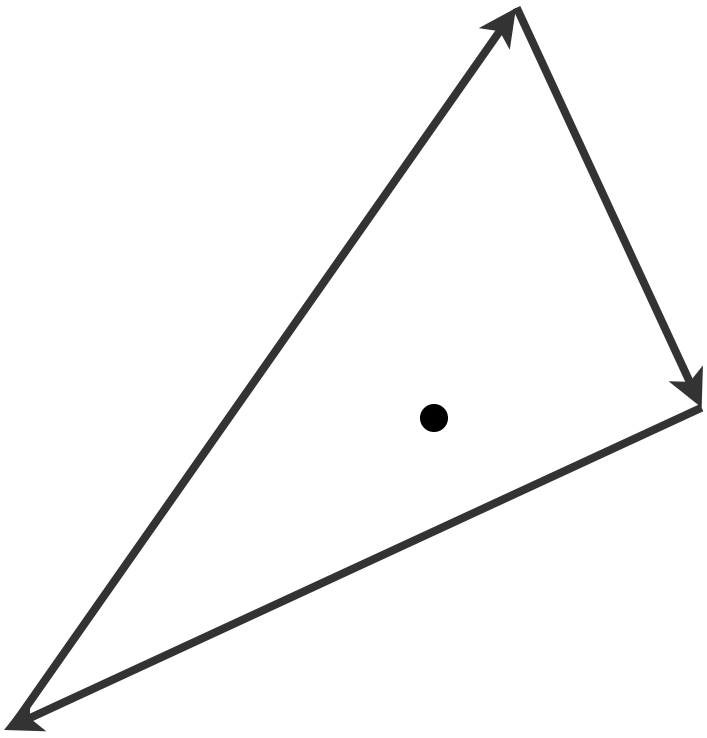
Декомпозиция



Декомпозиция



Декомпозиция



Необходимые сущности

- Прямоугольник
- Треугольник
- Вектор
- Точка

Точка


```
typedef float CoordinateType;


typedef struct {
    CoordinateType x;
    CoordinateType y;
} Point;
```

Вектор


```
typedef struct {
    Point begin;
    Point end;
} Vector
```

Вектор


```
typedef enum {
    SideRight = 0,
    SideLeft = 1
} Side;

Side vector_side(Vector v, Point p);
```

Вектор


```
CoordinateType sign(Vector v, Point p)
{
    return (p.x - v.end.x) * (v.begin.y - v.end.y)
            - (v.begin.x - v.end.x) * (p.y - v.end.y);
}

Side vector_side(Vector v, Point p)
{
    return sign(v, p) < 0;
}
```

Треугольник


```
typedef struct {
    Point a;
    Point b;
    Point c;
} Triangle;

bool triangle_contains_point(Triangle t, Point p);
```

Треугольник


```
bool triangle_contains_point(Triangle t, Point p)
{
    const Vector v1 = {t.a, t.b};
    const Vector v2 = {t.b, t.c};
    const Vector v3 = {t.c, t.a};

    const Side s1 = vector_side(v1, p);
    const Side s2 = vector_side(v2, p);
    const Side s3 = vector_side(v3, p);

    return s1 == s2 && s2 == s3;
}
```

Прямоугольник


```
typedef struct {
    Point a;
    Point b;
    Point c;
    Point d;
} Rectangle;

bool rectangle_contains_point(Rectangle r, Point p);
```

Прямоугольник


```
bool rectangle_contains_point(Rectangle r, Point p)
{
    const Triangle t1 = {r.a, r.b, r.c};
    const Triangle t2 = {r.c, r.d, r.a};

    return triangle_contains_point(t1, p)
        || triangle_contains_point(t2, p);
}
```

Пример


```
bool is_point_in_rectangle(
        float rx1, float ry1,
        float rx2, float ry2,
        float rx3, float ry3,
        float rx4, float ry4,
        float rx5, float ry5)
{
    bool t1_1 = ((rx5 - rx2) * (ry1 - ry2) - (rx1 - rx2) * (ry5 - ry2)) < 0;
    bool t1_2 = ((rx5 - rx3) * (ry2 - ry3) - (rx2 - rx3) * (ry5 - ry3)) < 0;
    bool t1_3 = ((rx5 - rx1) * (ry3 - ry1) - (rx3 - rx1) * (ry5 - ry1)) < 0;

    bool t2_1 = ((rx5 - rx3) * (ry1 - ry3) - (rx1 - rx3) * (ry5 - ry3)) < 0;
    bool t2_2 = ((rx5 - rx4) * (ry3 - ry4) - (rx3 - rx4) * (ry5 - ry4)) < 0;
    bool t2_3 = ((rx5 - rx1) * (ry4 - ry1) - (rx4 - rx1) * (ry5 - ry1)) < 0;

    return (t1_1 == t1_2 && t1_2 == t1_3) || (t2_1 == t2_2 && t2_2 == t2_3);
}
```

Модули

- В языке C нет поддержки модулей
- Независимые сущности обычно организовывают как пару файлов: заголовочный (.h) и файл реализации (.c)

Структура заголовочного файла

Файл Rectangle.h

```
#ifndef RECTANGLE_H
#define RECTANGLE_H

typedef struct {
    Point a;
    Point b;
    Point c;
    Point d;
} Rectangle;

bool rectangle_contains_point(Rectangle r, Point p);

#endif
```

Структура заголовочного файла

- Конструкция ifndef-define-endif — Include guards
- Цель: предотвращение повторого включения файла

See: ODR (One Definition Rule)

Файл реализации

Файл Rectangle.c

```
#include "Rectangle.h"



bool rectangle_contains_point(Rectangle r, Point p)
{
    ...
}
```

The Most Important Design Guideline

> *Make interfaces easy to use correctly and hard to use incorrectly*
>
> Scott Meyers – The Most Important Design Guideline

Возможности

> *Affordance (возможности) — в психологии восприятия и в области дизайна человеко-машинного взаимодействия означает свойства объекта, которые позволяют выполнить с ним те или иные действия.*

Be consistent

- `int fscanf(FILE *stream, const char *format, ...);`
- `int fgetpos(FILE *stream, fpos_t *pos);`
- `int fseek(FILE *stream, long offset, int whence);`

Be consistent

- `int fscanf(FILE *stream, const char *format, ...);`
- `int fgetpos(FILE *stream, fpos_t *pos);`
- `int fseek(FILE *stream, long offset, int whence);`

- `int fputc(int c, FILE *stream);`
- `FILE *freopen(const char *path, const char *mode, FILE *stream);`

Introduce new types


```
void print_date(int year, int month, int day);

print_date(3, 12, 2016);
```

Introduce new types


```
void print_date(Year year, Month month, Day day);

print_date((Year){2016}, (Month){12}, (Day){3});
```

The Most Important Design Guideline

Scott Meyers – The Most Important Design Guideline

https://youtu.be/5tg1ONG18H8