Object-oriented programming

based on the C++ language

2-nd semester

"Programming today is a race between software engineers striving to build bigger and better idiot-proof programs. and the Universe trying to produce bigger and better idiots. So far. the Universe is winning."
Rich Cook

lecture №1

Teacher:

Associate Professor, Department "Dynamics and strength of machines"

Ph.D. Vodka Alexey (a. 12)

Course:

  0.5 lecture per week
  2.5 labs per week

  exam

Literature

- Shildt The Theory and Practice of C ++. - SPb .: BHV, 1996
- R. Laforêt. Object-oriented programming in C ++. - St. Petersburg: Peter, 2003 - 928 p.
- **Pavlovskaya TA** C / C ++. language programming
  high level. -CPb.: Peter 20012003.
- **Pavlovskaya TA YA Shchupak** C ++. Object-Oriented Programming: Practice. - St. Petersburg: Peter, 2004.
- Stephen R. Davis. C ++for Dummies. Williams, Dialectics, 2001
- Harvey DeitelPaul Deitel. How to programC ++. Bean, 2003city
- Bjorn Stroustrup. C ++ programming language. Special edition, Bean - 2006. 1104 from.

Introduction to Object-Oriented Programming (OOP)

Constructors and destructors

procedural languages

- Development of object-oriented method, due to limitations of the other programming methods developed previously.
- C, Pascal, FORTRAN and others similar to them programming languages are classified as procedural languages. Each operator of a languageis an indicating the computer to perform some action, such as to take data from the user to perform certain actions and to bring with them result these actions on the screen. Programs written in procedural languages represent a sequence of instructions.

procedural languages

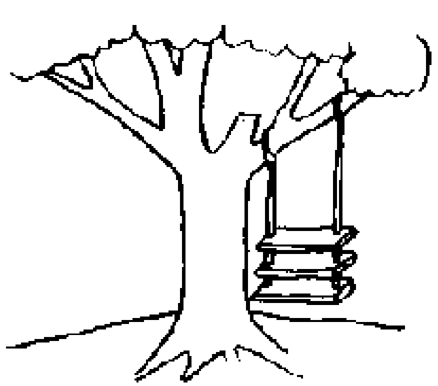
Data + Algorithm = Program

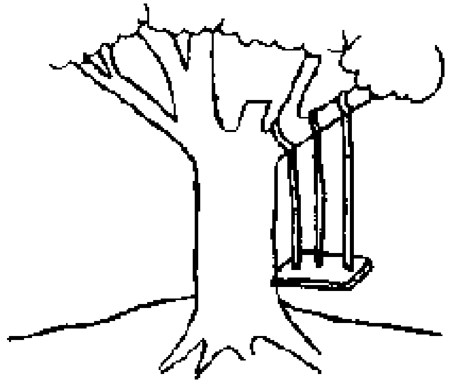
Division by function

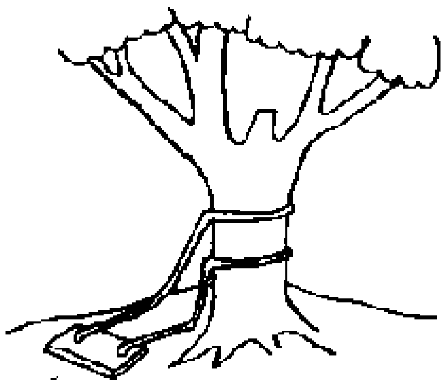
- When the program size is large, the list of commands becomes too cumbersome. A very small number of programmers able to holdhead over 500 lines of code, if the code is not divided na more small logic part. Function a means of facilitating the perception reading text programs.
- The program is built on the basis of procedural methods divided to functions, each of which in the ideal case performs some complete sequence of actions and has clearly expressed connection with other features

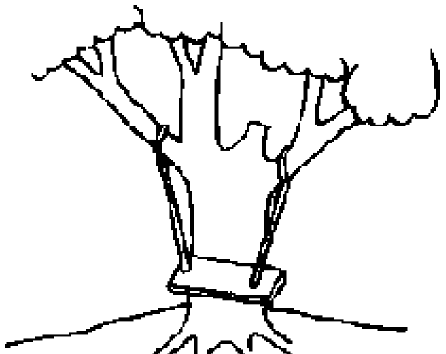
Disadvantages of structured programming

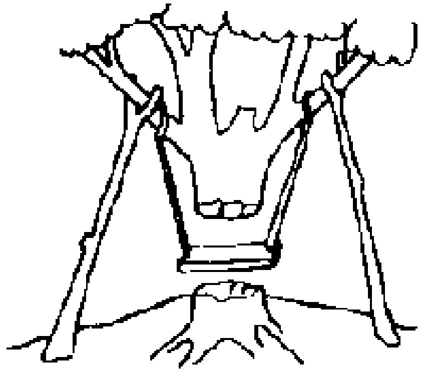
- In the ongoing process of growth and complexity of the programs became gradually come to light disadvantages of the structural approach to programming.
- Work on program project in the following ways: the task is more complicated than it seemed, project deadlines tolerated. More and more programmers are attracted to work that sharplyincreases costs. End of the newly transferred, and as a resultproject suffers collapse.

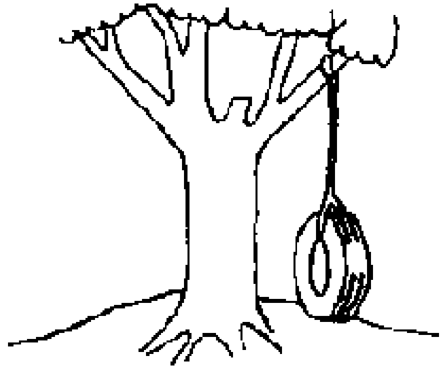

1. Как было предложено организатором разработки


2. Как было описано в техническом задании


3. Как было спроектировано ведущим системным специалистом


4. Как было реализовано программистами


5. Как было внедрено


6. Чего хотел пользователь

Two disadvantages of procedural programming

- absoluteness access functions to global data.
- Separation data and functionsbeing a structural approach, bad displays a picture real world.

- In a procedural program written, for example, in C, there is a two types of data.
- local data are in function and intended for Use this function only and not may be changed by other functions.
- If there is a need to share one and the same data multiple functions, the data must be declared global.

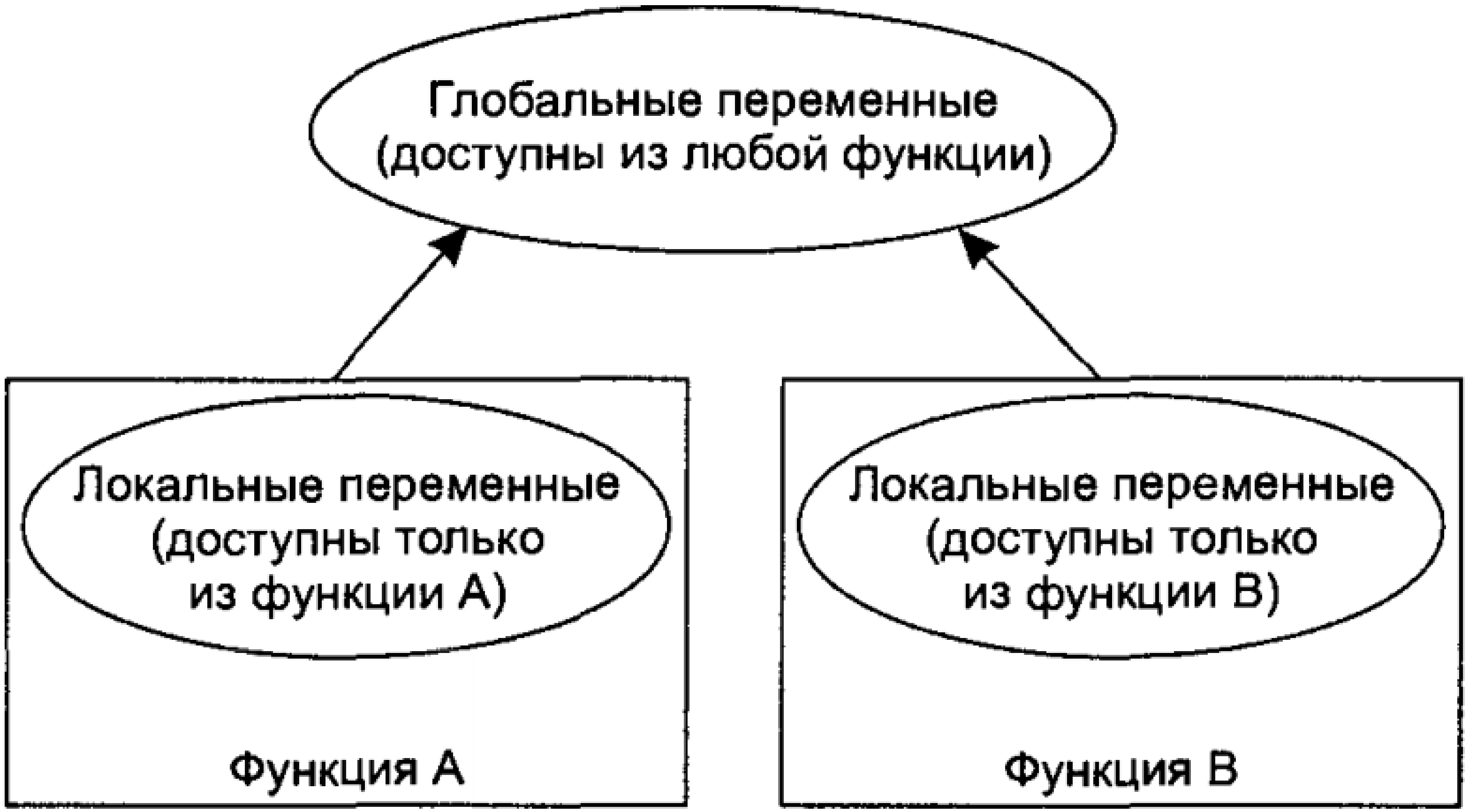


Рис. 1.1. Глобальные и локальные переменные

- Large programs typically contain a variety of functions and global variables. The problem of a procedural approach is that the number ofpossible linkages between global variables and functions can be very large.

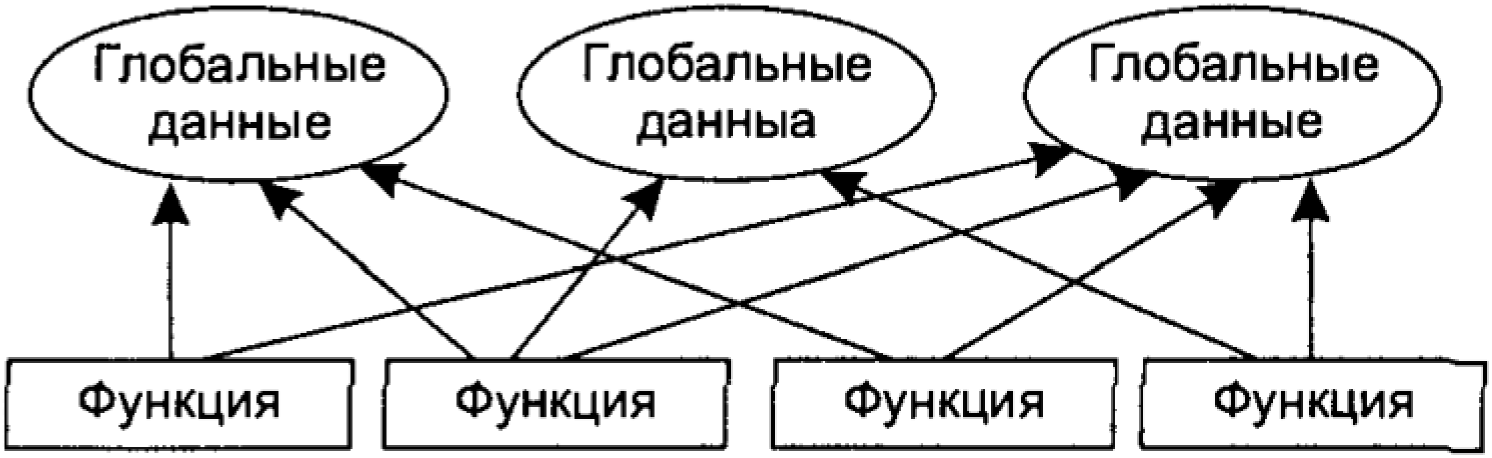


Рис. 1.2. Процедурный подход

A large number of connections between functions and generates data some problems:

- complicated structure programs
- IN program becomes difficult to make changes.

For example, if a developer program storage accounting decides to make the product code is not a 5-digit and 12-digit, it will be you must change the appropriate data type short on the long. It meansThat in all the functions that operate on the product code must be included change, allowing to process data type long. Thus, any change involves for a far-reaching consequences.

Modeling the real world

The second, more important, the problem is a procedural approach is that, that the department Information on the functions is of little use for display picture the real world. In the real world we have to deal withphysical objects, such as for example people or machines. these objectscan not be attributed any data or functions as the real thing represent a set of properties and behavior.

properties

Examples of properties (sometimes referred to as characteristics) for people may be eye color or place of work; for cars - engine powerquantity doors. Thus, the properties are equivalent to data objects inprogramsThey have a certain value, such as blue for the color eyes or 4 for the number of car doors.

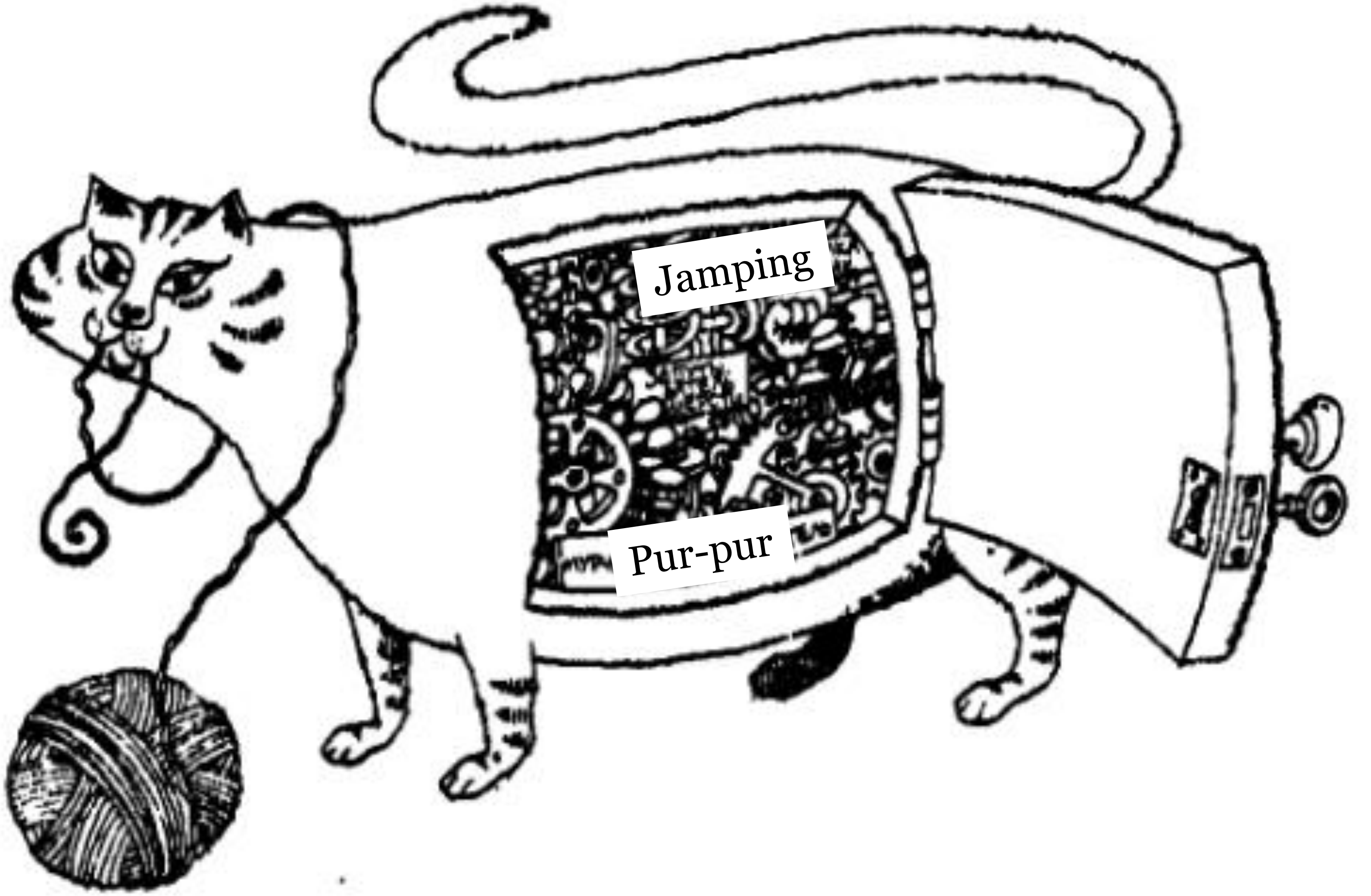
Behavior

Behavior - is a reaction to some object in response to an external stimulus. for instanceYour boss in response to a request for increase can answer "yes" or "no." If you hit the brakes the car, it would entail itstop. Response and stopping are examples of behavior. The behavior is similarfunction: you call a function to perform an action (eg, Display the account screen), and the function performs this action. Thus, any data taken separately or taken separately functionnot able to adequately display the real-world objects.

The object-oriented approach

- The basic idea of object-oriented approach is Union data and actions performed on that data into a single whole that It called object.
- object functions called in C ++, methods or Member functions normally designed to access the data object. If necessarycount any data object, you must call the appropriate method, which the performs read and returns the value. Direct access data impossible.

encapsulation- unification of data and methods to work with them in a single object. Encapsulation also implements the concealment data from the external influence that protectsthem from accidental change.

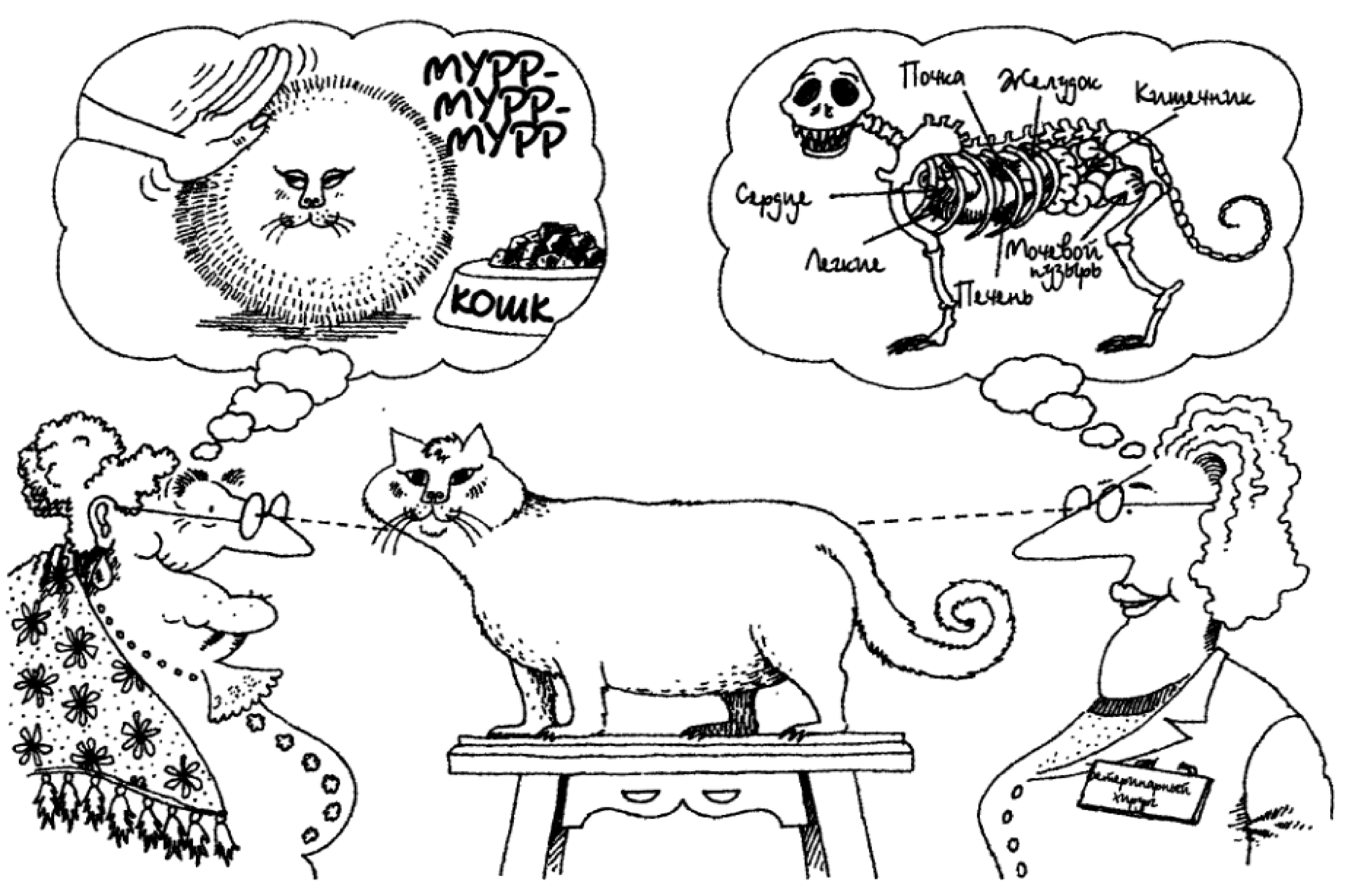


Инкапсуляция скрывает детали реализации объекта

abstracting

- abstracting - one of the main methods to cope with complexity.

Abstraction highlights the essential characteristics of a objectWhich distinguish it from all other kinds of objects and thus mannerClearly describes its conceptual boundaries in terms observer of.



Абстракция концентрирует внимание на существенных свойствах объекта с точки зрения наблюдателя

Object-oriented programming OOP (object-oriented programming - OOP) - this is method programmingBased on the presentation of the program in the form of together interacting objects, each of which is an instance of a particular class, and the classes are Member specific inheritance hierarchy.

Models of real objects (abstract) + behavior of objects (interaction) = Program

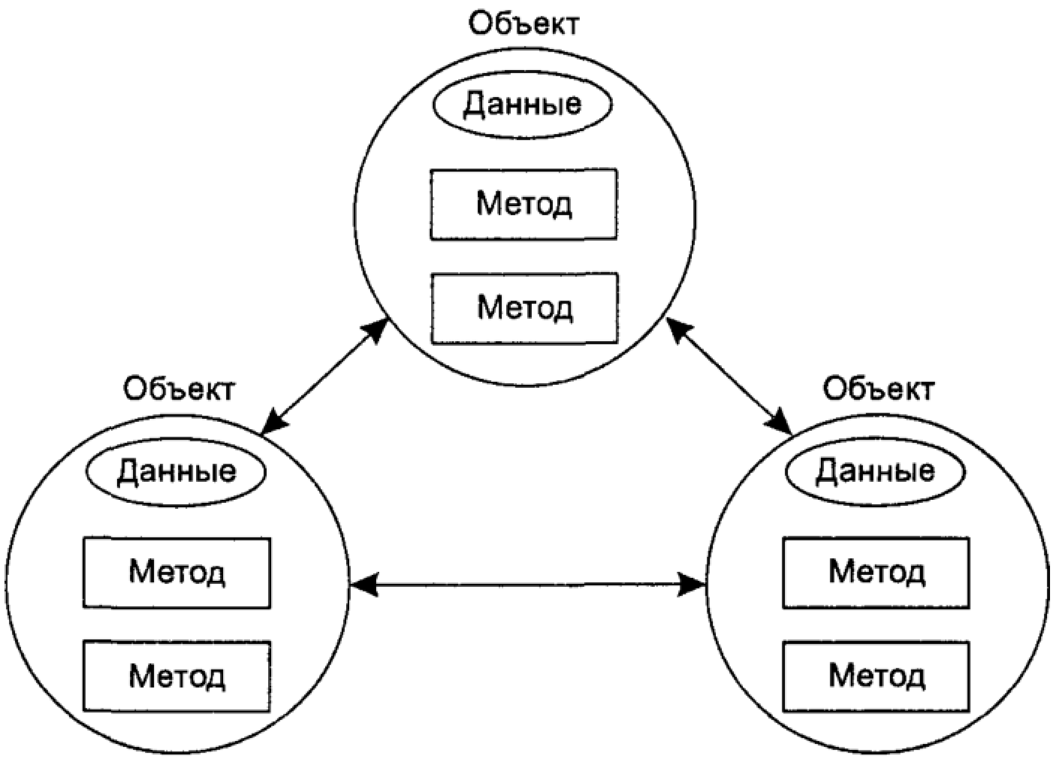


Рис. 1.3. Объектно-ориентированный подход

What could be the object (class)?

Physical objects.

- Cars in the simulation of traffic.
- schematics elements in the simulation circuit of an electric current.
- Countries to create economic model.
- Aircraft in the simulation of dispatching system.

UI elements

- Window.
- Menu.
- graphic objects (lines, rectangles, circles).
- Mouse, Keyboard, disk drives, printers.

data structures

- arrays.
- stacks.
- related lists.
- binary trees.

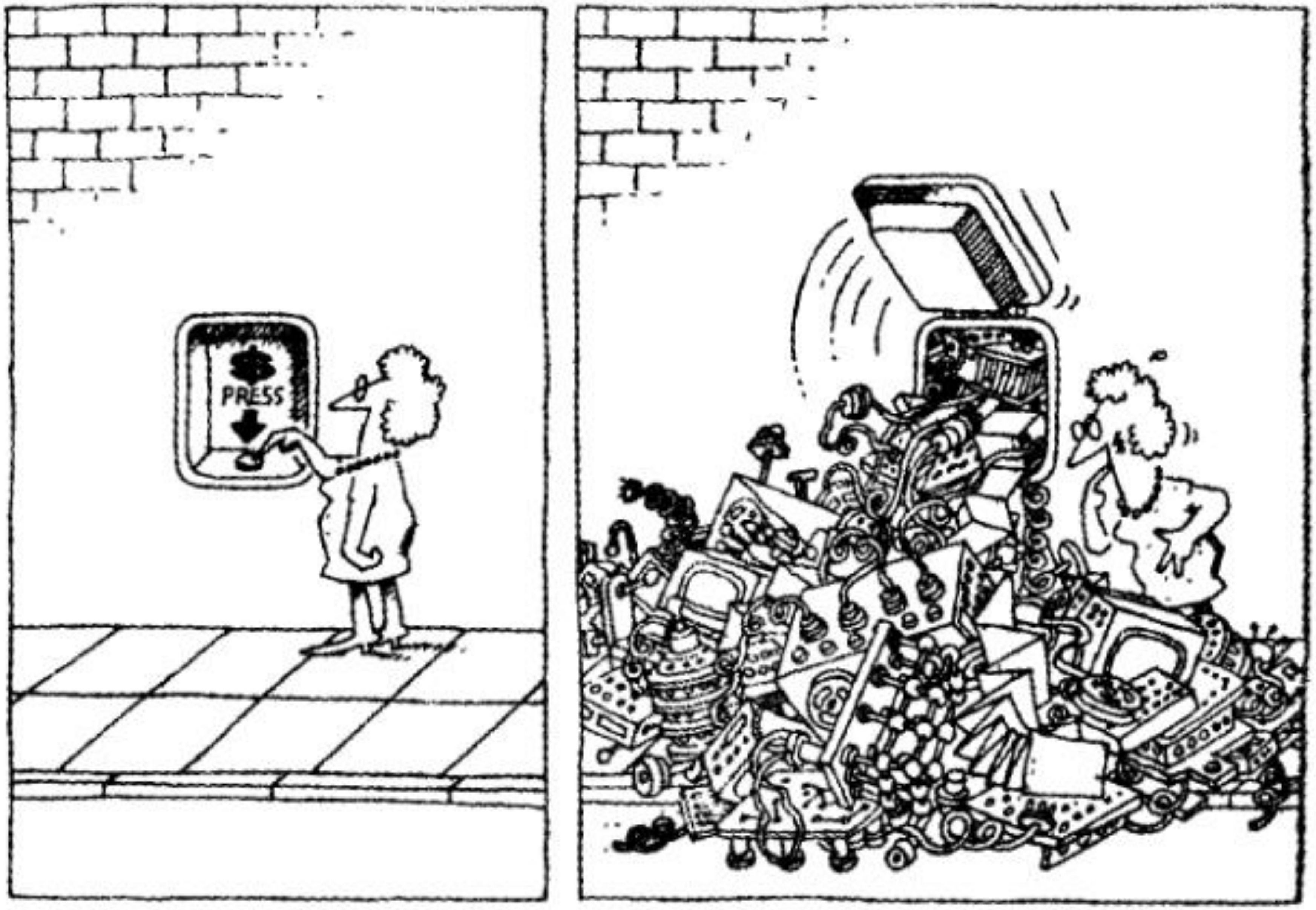
groups people
- Staff.
- Students.
- Buyers.
- Sellers.

Repositories data
- Inventory inventory.
- staff lists.
- Dictionaries.
- The geographical coordinates of cities in the world.

custom types data
- Time.
- The angles.
- Complex numbers.
- The point on the plane.

participants computer games
- Cars in the race.
- Positions in board games (checkers, chess).
- Animals in games related to wildlife.
- Friends and enemies in adventure games.

And other



Задача группы проектировщиков — создать иллюзию простоты

class

- A class is an abstract type, user-defined dataand It is a model of a real object in the form of data and functions for work with them.
- These classes are called fields (similar to the structure of fields), and functions class - methods. Fields and methods are called class members. Description class in the first approximation is as follows:

```
class <Name> {
[ private:]
<Description of hidden items>
public:
<Description available items>
}; // Description of the end with a
semicolon
```

private and public

- access specifiers **private** and **public** control the visibility of elements class. Elements described after the official wordprivateVisible only within the class. This type of access adopted in the default class.
- class interface described after qualifier public. The effect of any specifier spreads until the next specifier or end of class. Canset some sections private and public, Their order does not matter.

EXAMPLE class description

```
class monstr{
 int health, ammo; //class fields
 char * Name;
 public: // access specifier
 // designer
 monstr(int he = 100, int am = 10)
 { health = he; ammo = am;}
 void draw (int x, int y,
 int scale, int position);
 int get_health() {Return health;}
 int get_ammo() {Return ammo;}
};
```

void monstr:: draw (int x, int y, int scale, int position) {/ * Method body * /}

inline int monstr::get_ammo() {return ammo;}

Description of objects

```
  monstr Vasia;
  monstr Super(200, 300);
  monstr stado[100];
  monstr *beavis = new monstr (10);
monstr &butthead = Vasia;
```

Access to the elements of the object

```
int n = Vasia.get_ammo();
stado[5] .draw;
cout << beavis->get_health();
```

constructors

Constructor - class method, which is used to create and initialize an instance. constructor name is always the same name class.

- Designer does not return, even the type of void. It is impossible to get a pointer to the constructor.
- A class can have multiple constructors with different parameters for different types of initialization (the overload mechanism is used).
- The constructor is called with no parameters is called a default constructor.
- constructor parameters can be of any type, except for the same class. You can set the default settings. They may contain only one of the designers.

destructors

Destructor - class method, which is used to remove an instance of the class. Name destructor is made of symbol ~ (tilde) and behalf class. destructor not It has arguments and a return values.

- The destructor is called automatically when an object goes out of scope:
- for local objects - at the outlet of the block in which they are declared;
- for the global - as part of the exit procedure main;
- for the objects specified via pointers, the destructor is invoked implicitly when using the operation delete.

The destructor for the example should look

```
monstr:: ~monstr() {Delete [] name;}
```

Separation of interface from implementation

One of the most fundamental principles of developing good software is interface branch of the implementation:

at building program in C ++ each class definition is typically placed in a header file, and determine the methods of this class are placed in the source files with the same base name.

*The header file - **imya_klassa.h***

*source file - **imya_klassa.cpp***

Header files included (by #include) in each file that uses a class, but with the source files are compiled and linked with the file containing the main program (main).

***Example.** The order calls constructors and destructors.*

```
// File CreateAndDestroy.h
class CreateAndDestroy
{
public:
 CreateAndDestroy(int); //designer
 ~CreateAndDestroy(); // destructor
private:
 int data;
};
// ----------------End of File -----

//File CreateAndDestroy.cpp
#include <iostream>
using namespace std;
CreateAndDestroy::CreateAndDestroy(int value)
{
 data = Value;
 cout << "Object" << data << "constructor" << endl;
}
CreateAndDestroy:: ~CreateAndDestroy()
{ cout << "Object" << data << "destructor" << endl;
}
// ------- a file-End-----------
```

```cpp
// File main.cpp
#include <iostream>
#include "CreateAndDestroy.h"
using namespace std;
CreateAndDestroy first (1);
int main ()
{ cout << "main: Hello" << endl;
 CreateAndDestroy second (2);
 cout << "main: second created" << endl;
 CreateAndDestroy* third = new CreateAndDestroy(3);
 cout << "main: third created" << endl;
 delete third;
 cout << "main: third deleted" << endl;
 cout << "Exit from main" << endl;
 return 0;
}
```

```
1 Object constructor
main: Hello
2 Object constructor
main: second created
3 Object constructor
main: third created
Object 3 destructor
main: third deleted
Exit from main
Object 2 destructor
Object 1 destructor
```