Введение в объектно-ориентированное программирование

Почему нужно объектно-ориентированное программирование

- Рост объема и сложности программ
- Основная проблема: возможностей других методов программирования, таких как структурное программирование, не хватает

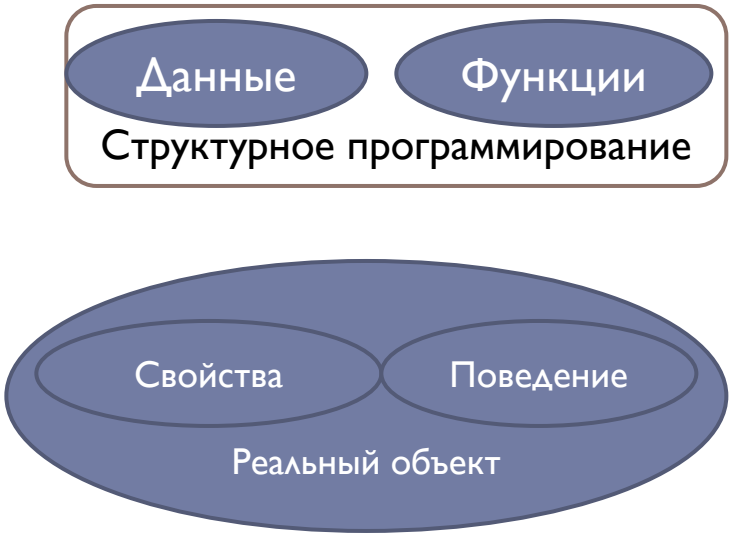
Структурное программирование

- Процедурные языки: Basic, Pascal, C
  - оператор представляет собой инструкцию компьютеру произвести некоторое действие: ввод данных, обработка данная, вывод данных
  - программа – последовательность инструкций
- Деление программы на модули, функции, отдельные инструкции
  - основа структурного программирования

Недостатки структурного программирования

1. Неограниченный доступ функций к глобальным данным

2. Плохое отображение картины реального мира

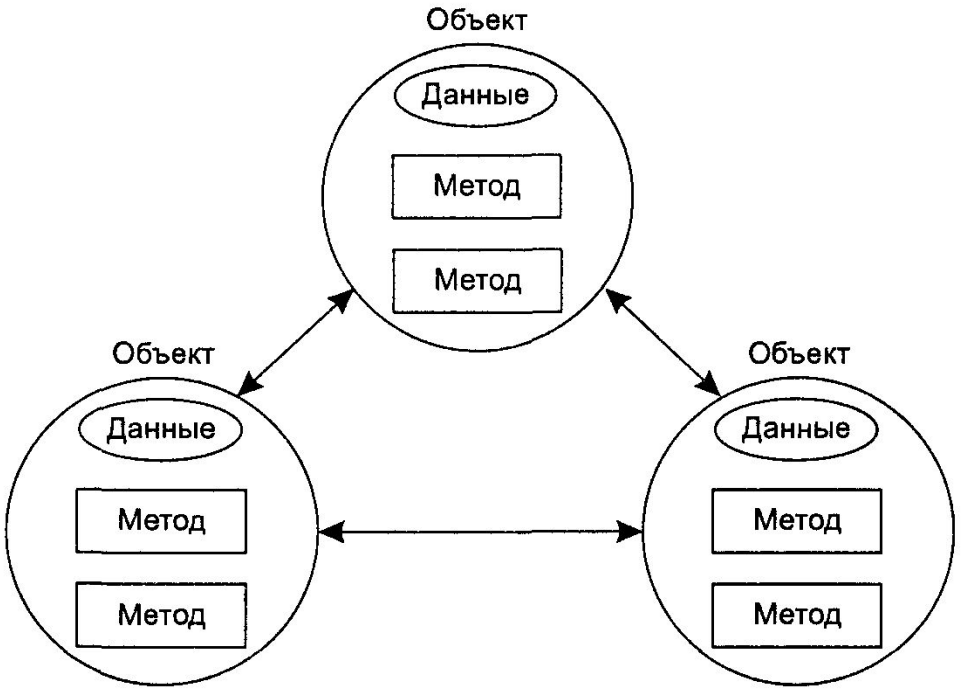


Проблемы:

1. Усложнение структуры программы
2. Сложности с внесением изменений в программу

Объектно-ориентированный подход

Объединение данных и действий, производимых над этими данными в единое целое, называемое объектом



Программа состоит из совокупности взаимодействующих между собой объектов

Принципы ООП

- Инкапсуляция
- Полиморфизм
- Наследование

Инкапсуляция -

механизм связывающий воедино код (методы) и данные, которыми он манипулирует, а также обеспечивающий их защиту от внешнего вмешательства и неправильного использования

Средство инкапсуляции – **объект**

Внутри объекта код и данные могут быть:

- открытыми – доступны из любой части программы
- закрытыми – доступны только из другой части этого же объекта

Объект - центральное понятие ООП

ЕСЛИ в структурном программировании - разбиение задачи на функции,
ТО в ООП - разбиение задачи на объекты

- Примеры объектов
  - Группы людей
    - Студент
    - Преподаватель
    - Пешеход (задача моделирования уличного движения)
  - Пользовательские типы данных
    - Время
    - Координаты точки на плоскости
  - Структуры данных
    - Массив
    - Список
  - Физические объекты
    - Автомобиль, светофор (задача моделирования уличного движения)
    - Элемент схемы (моделирование цепи электрического тока)
    - Страна (экономическая модель)
  - Элементы интерфейса

Объект Студент

Данные (атрибуты)

- Фамилия
- Имя
- Отчество
- Вуз
- Факультет
- Направление подготовки
- Курс
- Дата рождения
- Номер группы
- Зачетная книжка

Методы (операции, функции)

- Записать данные (фамилия, имя, отчество)
- Перевести на следующий курс (номер курса)
- Перевести на другое направление подготовки (направление подготовки)
- Вывести список студентов (номер группы)

Объект Зачетная книжка

Данные

- Номер зачетки
- Результаты промежуточной аттестации

Методы

- Записать номер зачетки
- Записать результат ПА
- Изменить результат ПА (результат ПА)

Объект Результат ПА

Данные

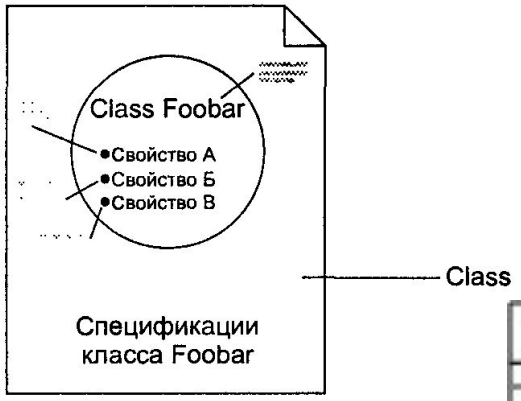
- Название дисциплины
- Оценка
- Дата сдачи
- Преподаватель

Методы

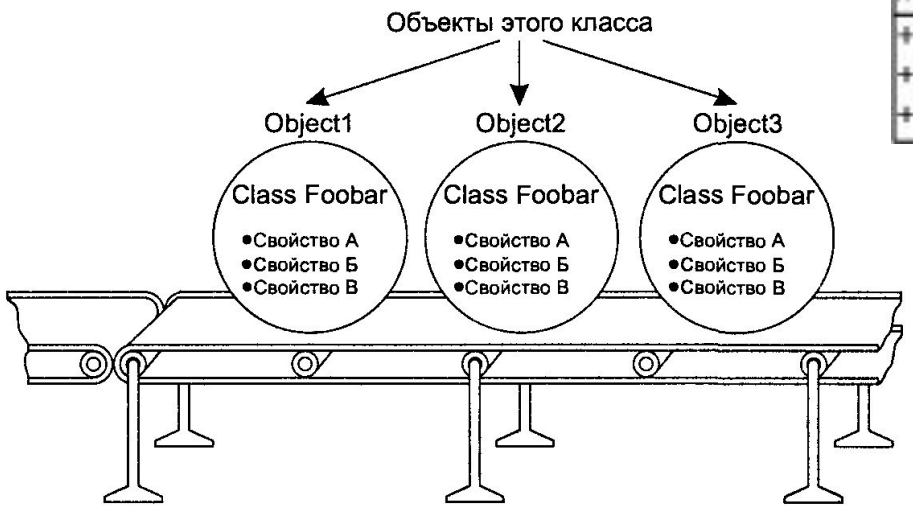
- Записать данные
- Изменить данные (данные)

Класс и его объекты

Новый тип объекта – это новый тип данных – это класс



Имя класса
-Имя закрытого атрибута 1 +Имя открытого атрибута 2 #Имя защищенного атрибута 3
+Имя метода 1() +Имя метода 2() +имя метода 3()



Полиморфизм -

атрибут, позволяющий с помощью одного интерфейса управлять доступом к целому классу объектов

«Один интерфейс – несколько методов»

Одно имя функции или оператора – разные варианты их определения, реализующие конкретные действия для каждого типа данных

Пример перегруженной функции:

```
int abs(int i);
double abs(double d);
long abs(long l);
```

Пример перегруженных операторов: <<, >>

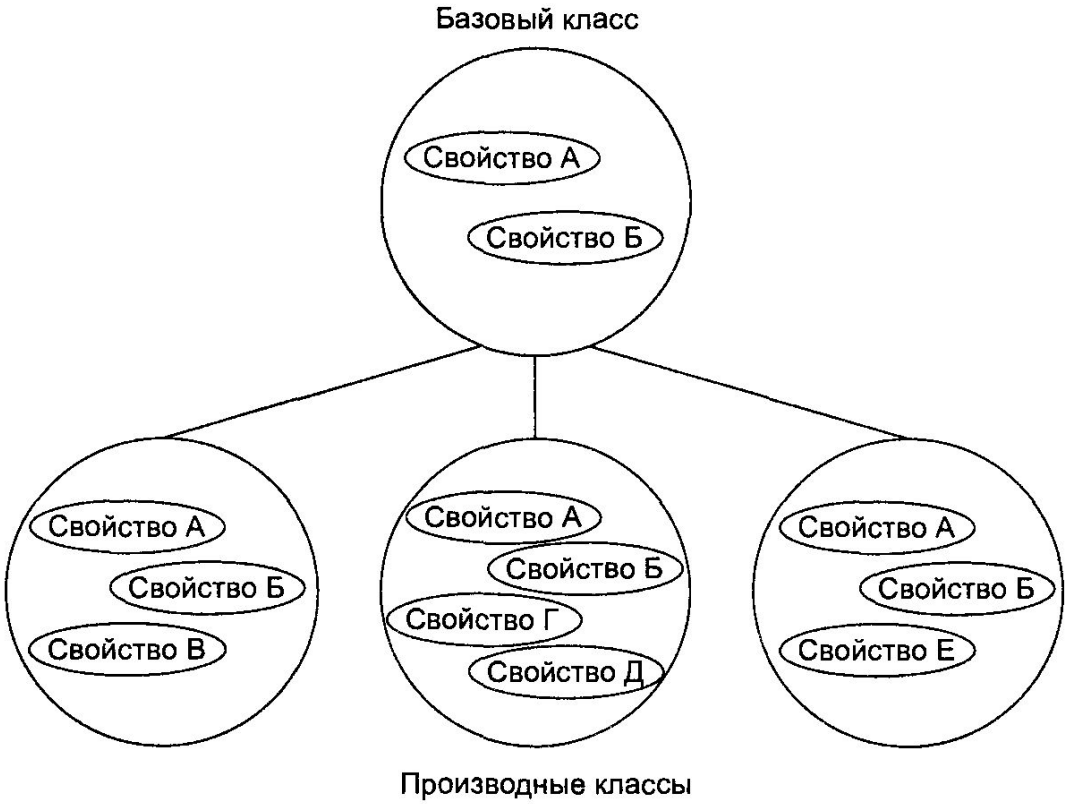
отношения между классами

Типы отношений:

- Отношение обобщения: наследование
- Отношение агрегации
- Отношение ассоциации

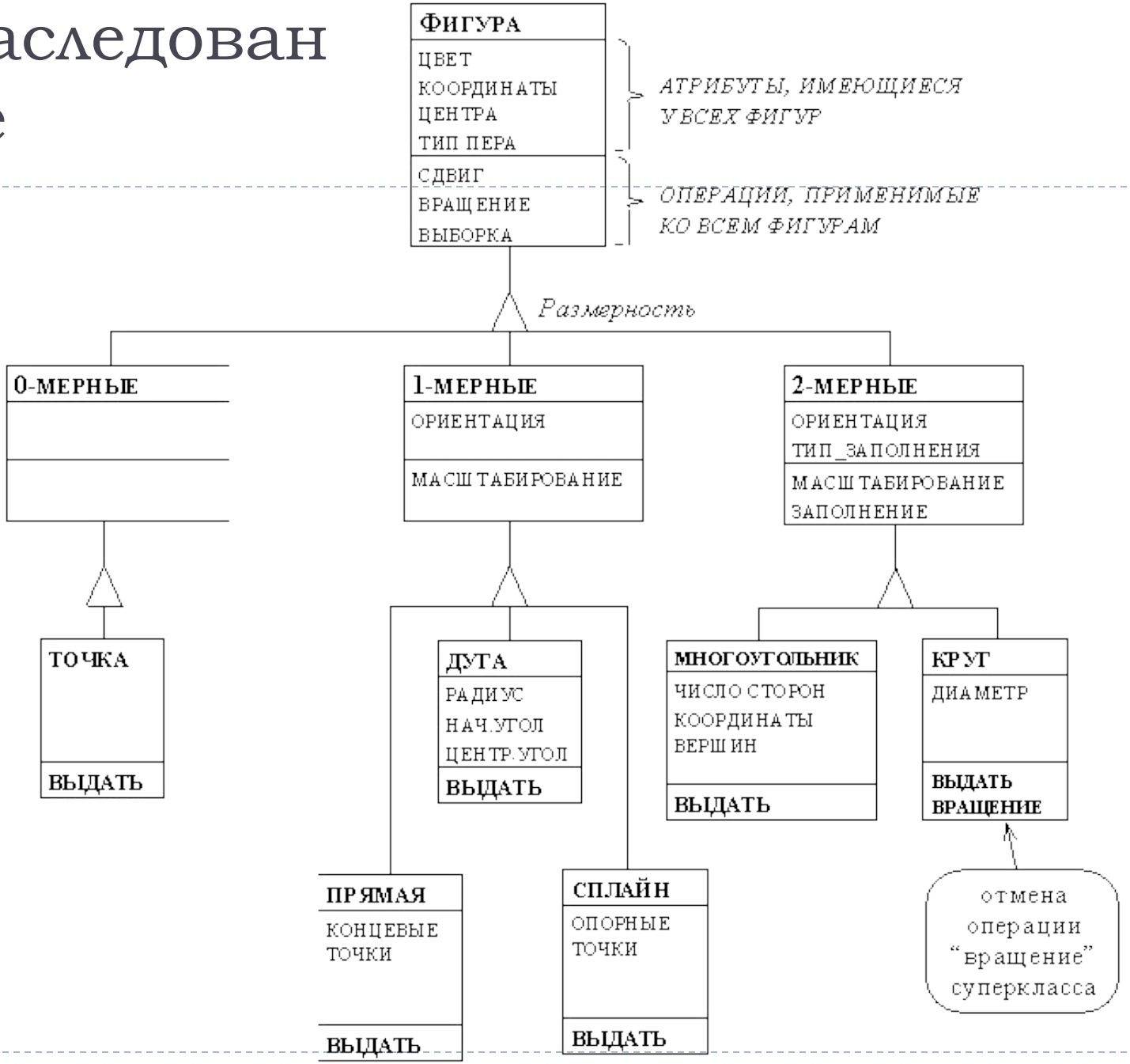
Отношение обобщения: наследование

Наследование – процесс, в ходе которого один объект может приобрести свойства другого

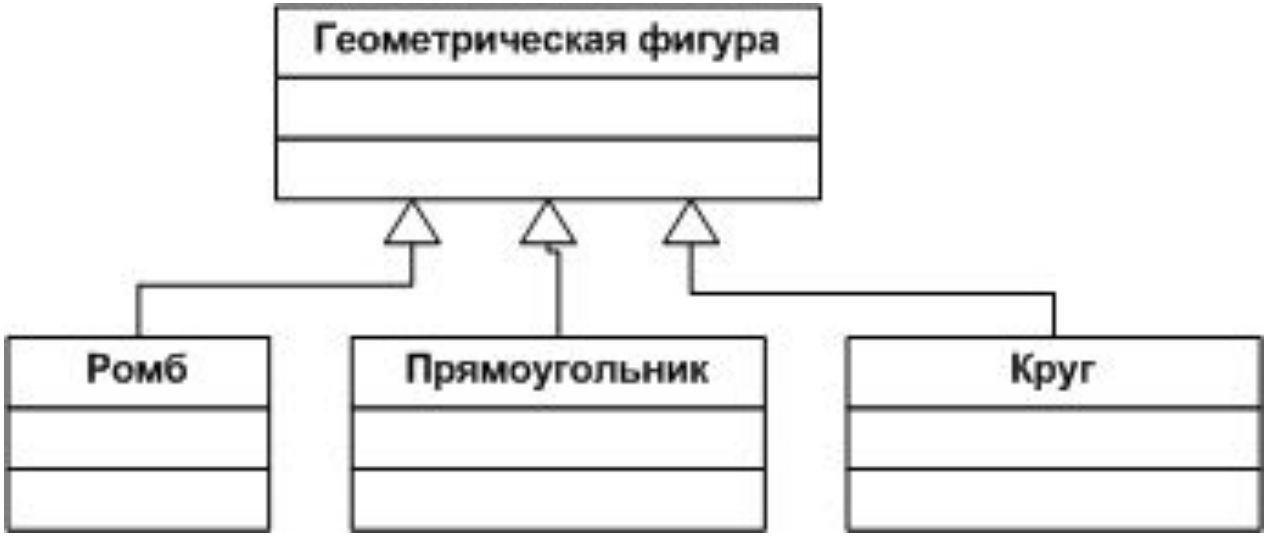


Третий принцип ООП

Наследование



Примеры изображения отношения обобщения



Отношение ассоциации

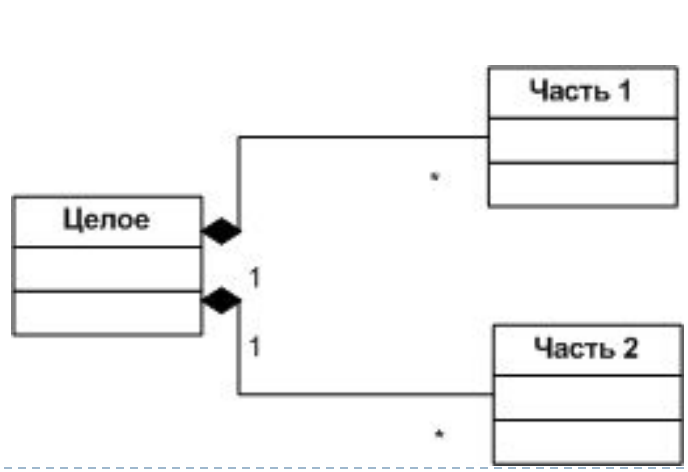
- Отношение ассоциации соответствует наличию некоторого отношения между объектами (классами)
- Самый простой случай данного отношения - бинарная ассоциация
  - Студент учится в Вуз
  - Страна имеет Столица
  - Программист является автором Проект

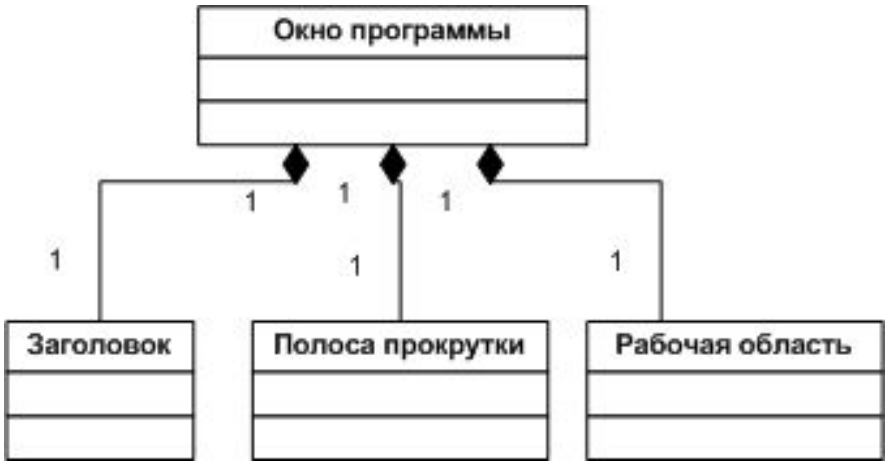
Отношение агрегации

- Один объект включает в себя в качестве составных частей другие объекты
- Данное отношение используется для описания структуры сложных систем, так как может представлять отношения типа «целое-часть«
- Отличие от отношения обобщения заключается в том, что части системы никак не обязаны наследовать ее свойства и поведение, поскольку являются вполне самостоятельными сущностями. Более того, части целого обладают своими собственными атрибутами и операциями, которые существенно отличаются от атрибутов и операций целого.

Отношение композиции

- Отношение композиции - частный случай отношения агрегации
- Это отношение служит для выделения специальной формы отношения «целое-часть", при которой составляющие части находятся внутри целого, то есть они не могут выступать в отрыве от целого
- При уничтожении объекта-целого уничтожаются и все объекты-части





Задание

- Какими отношениями связаны объекты: Студент, Зачетная книжка, Результат промежуточной аттестации?
- Изобразите классы и отношения между ними

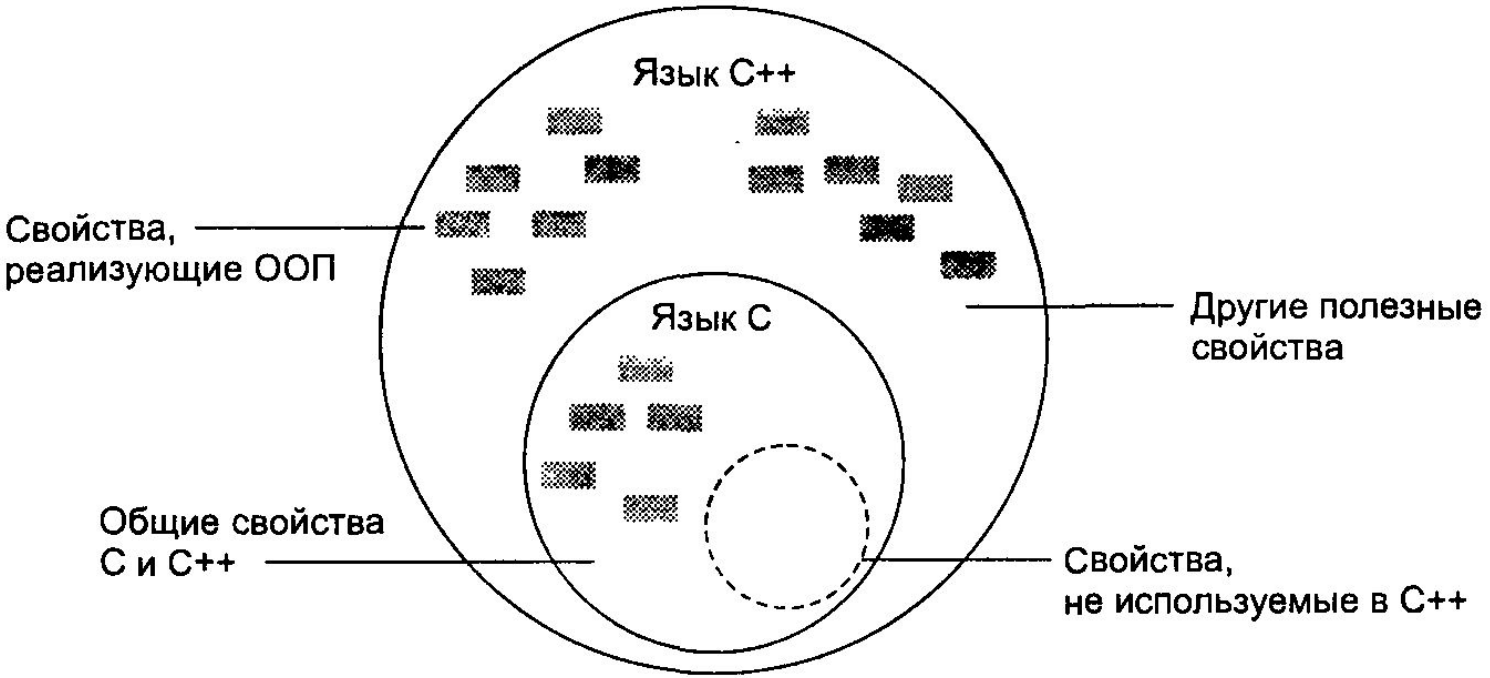
Задание на самостоятельную работу

Постановка задачи «Кошелек студента». Владелец кошелька может выполнить следующие действия с кошельком: добавить деньги в кошелек, взять деньги, пересчитать, посмотреть, дать деньги в долг. Источниками пополнения кошелька могут быть родители, также это может быть зарплата или стипендия.

Задание:

- Разработать и описать объектную модель Кошелька студента:
  - Модель должна быть максимально приближенна к реальной модели
  - Для представления модели использовать нотацию UML
  - Постараться использовать отношения иерархии, агрегации и ассоциации, отношения между классами именовать
  - В описании привести пояснения семантики методов класса

Язык C++


Язык C++
Свойства,
реализующие ООП
Язык C
Другие полезные
свойства
Общие свойства
C и C++
Свойства,
не используемые в C++

Структура программы

```
#include <iostream>
using namespace std;
// определения классов и методов

int main
{
// создание объектов классов
// вызов обычных функций
return 0;
}
// определения обычных функций
```

Объекты, функции и main()

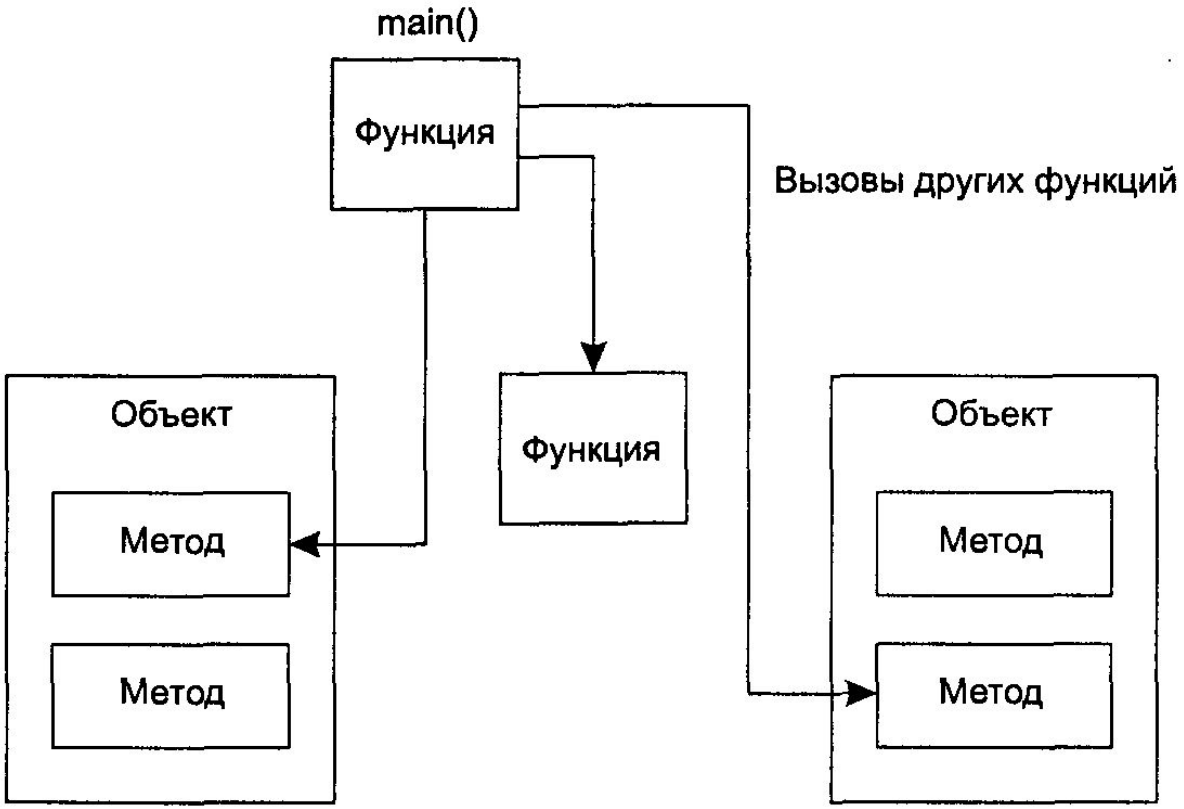


Рис. 2.1. Объекты, функции и main()

Классы и объекты

- Класс – логическая абстракция
- Объявление класса – объявление типа данных

- Объект – физическое воплощение логической абстракции
- Объявление объекта – объявление переменных типа class

Спецификаторы доступа:

public – открывает доступ к функциям и данным класса из других частей программы

private – закрывает доступ к функциям и данным класса из других частей программы, они доступны только члена данного класса

protected – необходим при наследовании классов

Общая форма объявления класса

```
class  имя_класса {
    закрытые данные и
  функции
спецификатор доступа:
    данные и функции
спецификатор доступа:
    данные и функции
…
спецификатор доступа:
    данные и функции
} список_объектов;
```

```
class  имя_класса {
    закрытые данные и
  функции
спецификатор доступа:
    данные и функции
спецификатор доступа:
    данные и функции
…
спецификатор доступа:
    данные и функции
};

имя_класса список_объектов;
```

Пример создания простых классов

- Разработать программу, которая вводит фактические данные из таблицы и выводит на экран таблицу, подобную той, которая находится в индивидуальном задании (включая заголовок и примечания)

Буддийские монастыри Японии периода Нара			
Название	Школа	Количество монахов	Площадь земли (га)
Тодайдзи	Т	220	368.8
Якусидзи	С	50	54.7
Дайандзи	Д	10	12.2
Примечание: Т - Тэндай; С - Сингон; Д - Дзедзицу			

Класс

```
class church {
    char *name; // все элементы данных private – по умолчанию
    char school;
    unsigned int count;
    float square;
public:
    void set(char *a,char b,unsigned int c, float d);
    void get(char *a, char &b, unsigned int &c, float &d);
    void show(void);
};//конец класса
```

Определения методов класса set и get

```
void church :: set(char *a,char b,unsigned int c,float d) {
name=new char[ ];
strcpy(name,a);
school=b;
count=c;
square=d;
}
 void church::get(char *a, char &b, unsigned int &c, float &d) {
delete [ ] a;
 a=new char [strlen(name)+1];
strcpy(a, name);
 b=school;
 c=count;
 d=square;
}
```

Ссылка – неявный указатель
Ссылка объявляется с помощью оператора &
Передача параметров в функцию с помощью ссылок
-передача адреса фактического параметра, то есть внутри функции идет работа напрямую с переменными b, c и d
Внутри функции оператор * - не нужен

Операторы ввода и вывода языка C++

Оператор ввода данных любого типа

>>

cin – идентификатор, связанный с клавиатурой

Оператор вывода выода данных

<<

cout – идентификатор, связанный с экраном

Пример

```
float a; int b; char str[10];
cout << введите два числа;
cin >> a >> b;
cout >> введите строку;
cin >> str;
cout << “a= “<< a << “  “ << “b = “<< b << “  “ << str;
```

Определение метода show(void)

```
void church::show(void) {
cout << name <<" ";
cout << school <<" ";
cout << count <<" ";
cout << square <<" ";
}
```

Определение функции main()

```
int main() {
char *n=new char[100];
char t;
unsigned int s;
float h;
short i;
church obj[5]; // создается массив объектов класса  church
cout<<"Работа функции SET!\n";
for (i=0; i<5; i++) {
cout<<"Название, Школа, Количество монахов, Площадь земли: \n";
cout<<"\n==>"; cin>>n;
cout<<"\n==>"; cin>>t;
cout<<"\n==>"; cin>>s;
cout<<"\n==>"; cin>>h;
obj[i].set(n,t,s,h); // вызывается метод класса set(n,t,s,h); , который
   записывает считанные с клавиатуры данные в поля объектов
}
```

```
cout<<"Работа функции SHOW!\n";
cout<<"Название, Школа, Количество монахов, Площадь земли: \n";
for (i=0; i<5; i++)  {
obj[i].show();
cout<<"\n";
  }
cout<<"Работа функции GET \n";
cout<<"Название, Школа, Количество монахов, Площадь земли: \n";
for(i=0; i<5; i++)  {
obj[i].get(n, t, s, h);  // получаем значения атрибутов объектов
cout << "Название: "<< n << " \n";
cout << «Школа:“<<t << " \n";
cout << «Количество монахов:“ << s << " \n";
cout << «Площадь земли:“ << h << " \n";
}
getch();
delete[] n;
return 0;
}
```

Задание

- Создайте класс Int, имитирующий стандартный тип int. Класс имеет одно поле типа int и методы:
  - метод, инициализирующий поле целым числом
  - метод, выводящий значение поля на экран
  - метод, складывающий два значения типа Int
- В программе должно быть создано три объекта класса Int. Два из которых будут инициализированы.
- Сложите два инициализированных объекта и присвойте результат третьему, а затем отобразите результат на экране

```
#include "iostream"
#include "stdlib.h"
using namespace std;
class Int {
      int x;
public:
      void setX(int a);
    void show();
      int summa(Int obj1, Int obj2)
    { return obj1.x+obj2.x;}
    };
   void Int::setX(int a){ x=a; }
   void Int::show()
   { cout << x; };
void main()
```

Задание на самостоятельную работу

Постановка задачи «Кошелек студента». Владелец кошелька может выполнить следующие действия с кошельком: добавить деньги в кошелек, взять деньги, пересчитать, посмотреть, дать деньги в долг. Источниками пополнения кошелька могут быть родители, также это может быть зарплата или стипендия.

Задание:

- Разработать классы, реализующие объектную модель «Кошелек студента»:
  - На данном этапе пока создайте простые классы, которые не связаны отношением наследования
  - Создать данные и методы, позволяющие записать информацию в поля класса, получить информацию из полей класса и вывести на консоль

ВНИМАНИЕ! Реализуем пока то, что умеем, все остальные возможности будем добавлять в классы по мере изучения