

Технологии и методы программирования

Часть 2

Ст. преподаватель
кафедры ПИВТ
Воронцова И.О.

2020 год

Объектно-ориентированный подход в программировании

- Объектная декомпозиция задачи.
- Объект имеет:
 - имя,
 - набор свойств,
 - состояние,
 - модель поведения.
- Объекты взаимодействуют между собой с помощью своих интерфейсов.

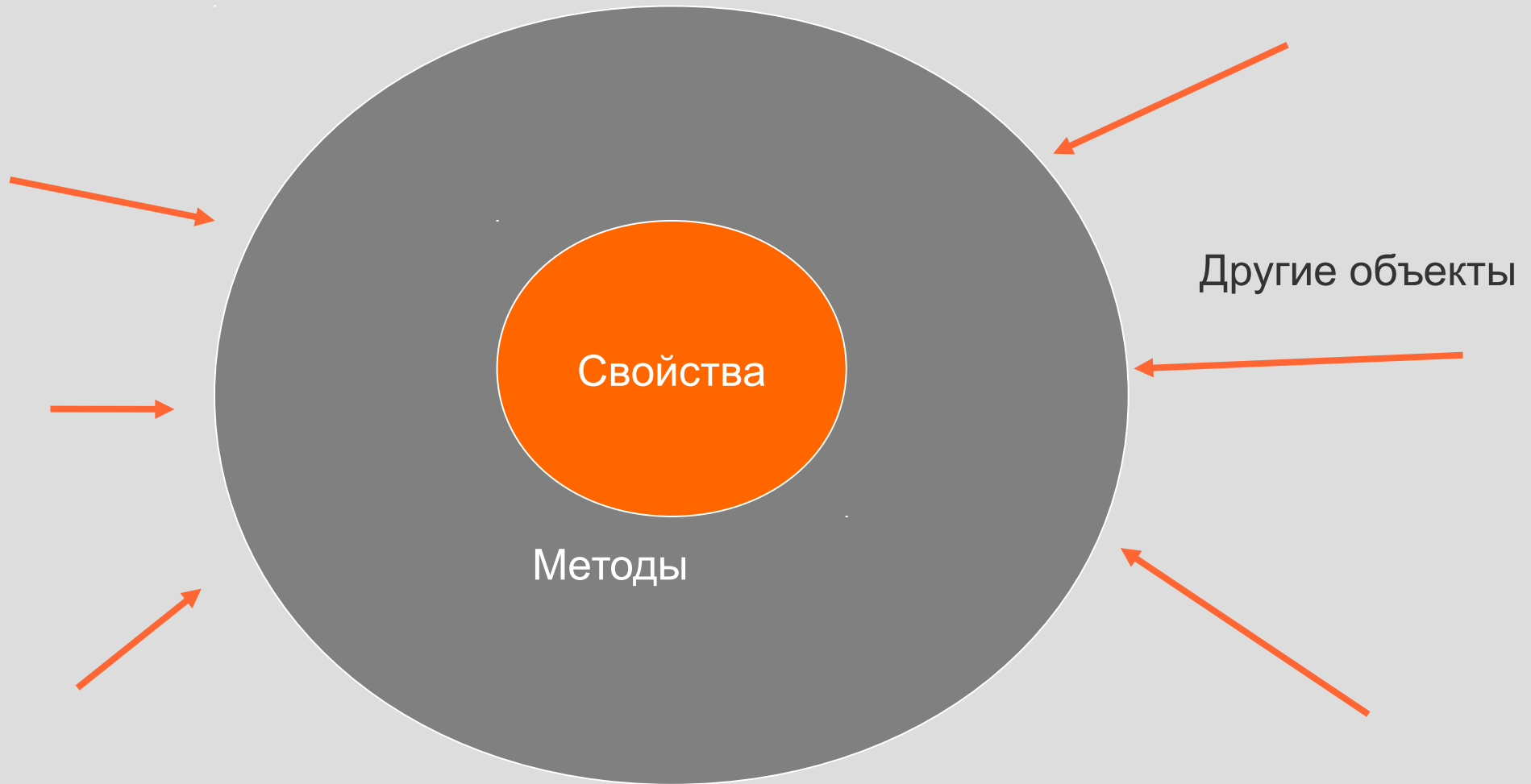
Понятия объектно-ориентированного программирования (ООП)

- Объект (object).
- Класс (class).
- Свойство (property):
 - не изменяющиеся в течение времени
 - изменяющиеся в течение времени (переменные состояния).
- Метод (method).
- Состояние (state):
 - переходы между состояниями.
- Сообщение (message).

Принципы ООП

- Инкапсуляция.
- Полиморфизм.
- Наследование.

Инкапсуляция



Класс в C++

- Объявление класса

```
class MyClass
{
public:
    // доступно всем
private:
    // доступно только данному классу
protected:
    //доступно классу и его наследникам
};
```

- Создание объекта

```
MyClass myObject;
```

Объявление класса

```
class Person
{
public:
    Person();
    std::string getName();
    void setName(std::string val);
    int getYear();
    void setYear(int val);
private:
    std::string name;
    int year;
};
```

Реализация функций-членов класса

```
Person::Person ()  
{  
    //Конструктор по умолчанию  
}
```

ИЛИ

```
Person::Person ()  
{  
    name = "Noname";  
    year = 1992;  
}
```


Реализация функций-членов класса (2)

```
void Person::setName(std::string nval)
{
    name = nval;
}
```

```
void Person::setYear(int val)
{
    year = val;
}
```

Реализация функций-членов класса (3)

```
std::string Person::getName()  
{  
    return name;  
}
```

```
int Person::getYear()  
{  
    return year;  
}
```

Программа с использованием объекта

```
int main(int argc, char** argv)
{
// Вызов конструктора по умолчанию
    Person p;
    p.setName("Anita");
    p.setYear(1978);
    cout<<p.getName()<<" "<<
        p.getYear()<<endl;
    return 0;
}
```

Конструирование класса, моделирующего работу устройства

Устройство: принтер.

Переменные, определяющие состояние устройства:

- наличие электропитания (да,нет),
- наличие бумаги (да, нет),
- текущее действие принтера (печатает, не печатает).

Методы:

- включить/выключить,
- добавить бумагу,
- убрать бумагу,
- отправить задание на печать,
- остановить печать,
- просмотреть текущее состояние принтера.

Таблица состояний принтера

Питание	Бумага	Действие принтера	Состояние принтера:
0	0	0	не включен, нет бумаги, не печатает
0	1	0	не включен, бумага есть, не печатает
1	0	0	включен, но нет бумаги, не печатает
1	1	0	включен, есть бумага, не печатает — ГОТОВ К РАБОТЕ
1	1	1	печатает

Объявление класса Printer (файл Printer.h)

```
class Printer {  
public:  
    Printer();  
    ~Printer();  
    void on_off();  
    void set_print();  
    void stop_print();  
    void paper_out();  
    void put_paper();  
    void show();  
private:  
    int is_on;  
    int is_print;  
    int has_paper;  
};
```

Реализация методов класса Printer (1) (файл Printer.cpp)

```
#include "Printer.h"
#include <iostream>

using namespace std;

Printer::Printer() : is_on(0),
    is_print(0), has_paper(0) {
}

Printer::~Printer() {
}
```

Реализация методов класса Printer (2)

```
void Printer::on_off() {
    is_on = !is_on;
    is_print = 0;
}

void Printer::set_print() {
    if (is_on&&has_paper)
        is_print=1;
}

void Printer::stop_print() {
    is_print=0;
}
```


Реализация методов класса Printer (3)

```
void Printer::paper_out() {
    has_paper=0; is_print=0;
}

void Printer::put_paper() {
    has_paper=1;
}

void Printer::show() {
    if(is_on) cout << "включен"<< endl;
    else cout << "выключен"<< endl;
    if(has_paper) cout <<"есть бумага" << endl;
    else cout <<"бумага закончилась" << endl;
    if (is_print) cout <<"печатаю"<<endl;
    else cout << "ГОТОВ ВЫПОЛНИТЬ ВАШЕ ЗАДАНИЕ!"
    <<endl;
}
```

Главная функция (файл main.cpp)

```
#include <iostream>
#include "Printer.h"

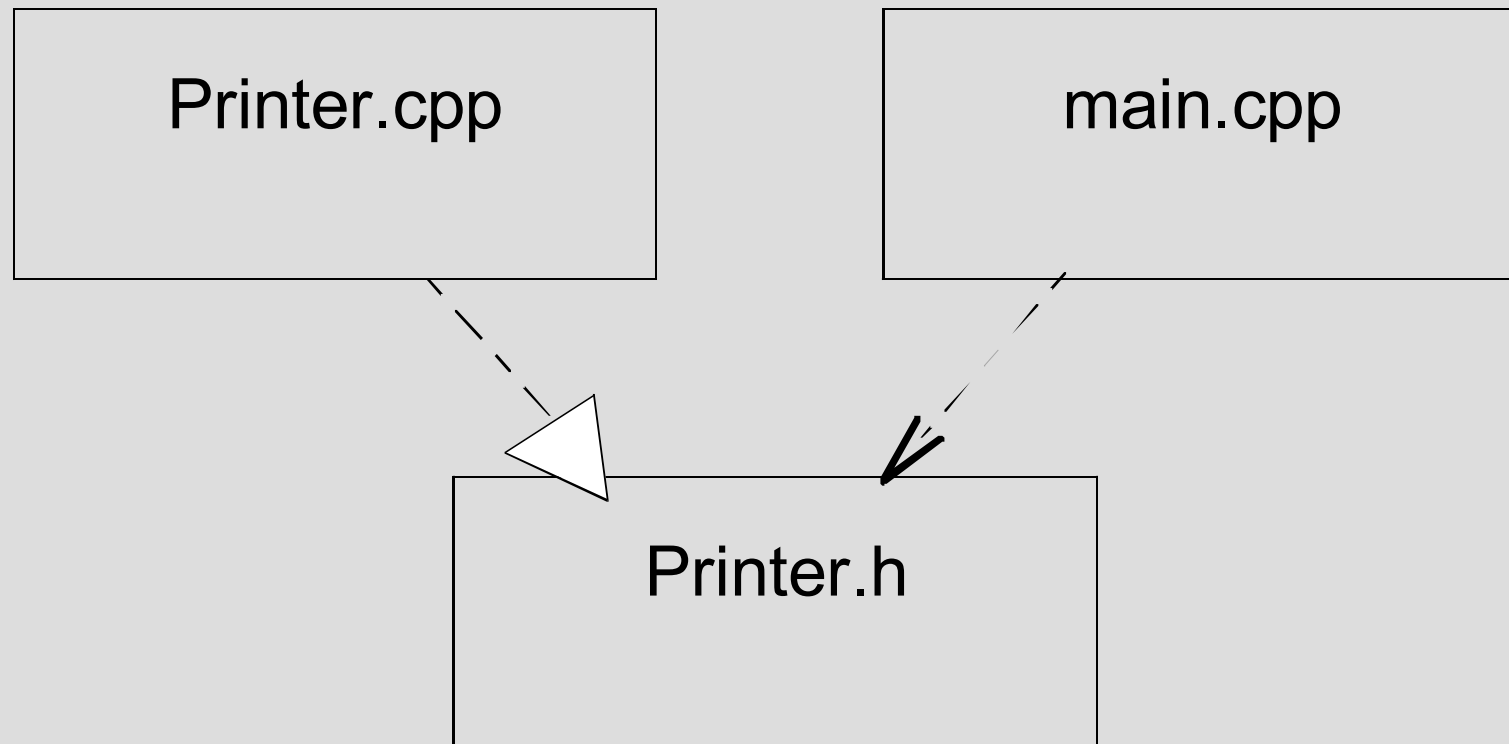
using namespace std;

int main()
{
    Printer pr;
    int command, end=1;
    pr.show();
```

Главная функция (продолжение)

```
while (end) {
    cout << " Введите команду ->" ;
    cin >> command;
    switch (command) {
        case 1: pr.on_off(); pr.show(); break;
        case 2: pr.set_print(); pr.show(); break;
        case 3: pr.stop_print(); pr.show(); break;
        case 4: pr.paper_out(); pr.show(); break;
        case 5: pr.put_paper(); pr.show(); break;
        default: end=0;
    }
}
return 0;
} //конец main()
```

Диаграмма файлов (компонентов) проекта



Перегрузка функций

- Пример – стандартная математическая функция возведения в степень pow()

```
long double pow(long double,int);
```

```
long double pow(long double,long double);
```

```
float pow(float,int);
```

```
float pow(float,float);
```

```
double pow(double,int);
```

```
double pow(double,double);
```

Перегрузка методов класса

- Пример – класс «Графический редактор»

```
// Объявление класса
class Draw
{
    char* message; // private по умолчанию
public:
    Draw ();
    ~Draw ();
    void paint (); // Рисует текст
    void paint (int, int,int,int); // Рисует
                                // прямоугольник
    void paint (int, int, int); // Рисует
                                //окружность
} ;
```

- Пример – класс «Графический редактор»

```
// Реализация методов (1)
Draw::Draw ()
{
    message = new char [10];
    strcpy (message, "text");
}
Draw::~Draw ()
{
    delete[ ] message;
    // message = NULL; - разрушается весь объект!
}
```

- Пример – класс «Графический редактор»

```
// Реализация методов (2)
void Draw::paint ()
{
    std::cout << message << std::endl;
}
void Draw::paint(int cx1, int cy1, int cx2, int
    cy2)
{
    std::cout<<"Рисуем прямоугольник" <<std::endl;
}
void Draw::paint (int cx1, int cy1, int r)
{
    std::cout << "Рисуем окружность" << std::endl;
}
```


- Пример – класс «Графический редактор»

```
// Функция main ()
int main ()
{
    Draw draw;
    draw.paint (); // Рисует текст
    draw.paint (1,1,10,15); // Прямоугольник
    draw.paint (5,5,10); // Окружность
    return 0;
}
```

Виды конструкторов

- Конструктор по умолчанию
- Конструктор с параметрами
- Копирующий конструктор

```
class Person {  
    char* name;  
    int year;  
public:  
    Person (); // Конструктор по умолчанию  
    Person (char*,int); //Конструктор с параметрами  
    Person (const Person&); // Копирующий  
                                // конструктор  
    ~Person (); // Деструктор  
    // Другие методы  
};
```

Конструктор по умолчанию

- Реализация

```
Person::Person ()  
{  
}
```

или

```
Person::Person ()  
{  
    name = new char[7];  
    strcpy(name, "Noname");  
    year = 0;  
}
```

- Вызов конструктора по умолчанию

```
Person myPerson;
```

Конструктор с параметрами

- Реализация

```
Person::Person (char* n, int y)
{
    name = new char[strlen(n)+1];
    strcpy (name, n);
    year = y;
}
```

- Вызов конструктора с параметрами

```
Person myPerson ("Ann", 1990);
```

Конструктор с параметрами по умолчанию

- Реализация конструктора

```
Person::Person (char* n="Noname", int y=0)
{
    name = new char[strlen(n)+1];
    strcpy (name, n);
    year = y;
}
```

- Вызов конструктора

```
Person myPerson ("Ann", 1990); // или
Person myPerson;
```

Конструктор со списком инициализации

- Реализация конструктора

```
Person::Person (char* n,  
int y) : name (new char[strlen(n+1)]), year (y)  
{  
    //Если свойство name — указатель  
    strcpy (name, n);  
}
```

- Вызов конструктора

```
Person myPerson ("Ann", 1990);
```

Копирующий конструктор

- Реализация

```
Person::Person (const Person& p) {  
    name = new char[strlen(p.name)+1];  
    strcpy (name, p.name);  
    year = p.year;  
}
```

- Вызов копирующего конструктора

```
//исходный объект
```

```
    Person myPerson ("Ann", 1990);
```

```
//его копия
```

```
    Person newPerson (myPerson);
```

Деструктор

- Реализация

```
// по умолчанию
```

```
Person::~Person ()
```

```
{
```

```
}
```

```
// для рассматриваемого примера
```

```
Person::~Person ()
```

```
{
```

```
    delete [ ] name;
```

```
}
```

- Вызов деструктора – явный вызов не требуется.