Семинар 5

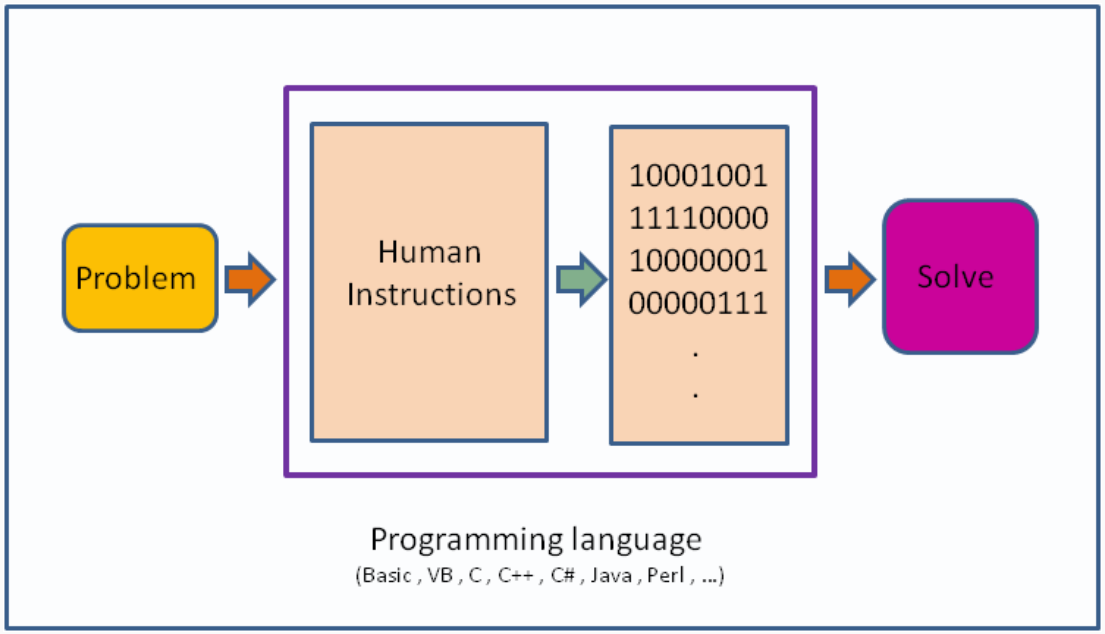
Введение в C++. Классы.

Бирюков Владимир

МФТИ

ООП. Классы

Программирование



Парадигмы программирования:

- **Процедурное** – отражение архитектуры традиционных ЭВМ. C, Pascal, C++, python
- **Объектно-ориентированное** – представление программы в виде совокупности объектов. Java, C#, C++, python
- **Функциональное** – процесс вычисления как вычисление некой математической функции. Haskell

ООП

ООП — методология программирования, основанная на представлении программы в виде совокупности объектов, каждый из которых является экземпляром определенного класса, а классы образуют иерархию наследования.

Основные принципы:

- Инкапсуляция
- Наследование
- Полиморфизм

ООП в стиле С

```
struct Monster
{
    float x, y, z;
    int health, is_alive, power;
};

void hurt_monster(Monster* m, int damage)
{
    m->health -= damage;
    if (m->health < 0)
        m->is_alive = 0;
}
void heal_monster(Monster* m, int heal_power)
{
    m->health += heal_power;
}
```

ООП в стиле C++

```
struct Monster
{
    float x, y, z;
    int health, is_alive, power;

    void hurt(int damage)
    {
        m->health -= damage;
        if (m->health < 0)
            m->is_alive = 0;
    }
    void heal(int heal_power)
    {
        m->health += heal_power;
    }
};
```

Инкапсуляция

```
class Monster
{
private:
    float x, y, z;
    int health, is_alive, power;
public:
    void hurt(int damage)
    {
        m->health -= damage;
        if (m->health < 0)
            m->is_alive = 0;
    }
    void heal(int heal_power)
    {
        m->health += heal_power;
    }
};
```

ООП в стиле C++

```
struct Monster
{
    float x, y, z;
    int health, is_alive, power;
    void hurt(int damage);
    void heal(int heal_power);
};
void Monster::hurt(int damage)
{
    m->health -= damage;
    if (m->health < 0)
        m->is_alive = 0;
}
void Monster::heal(int heal_power)
{
    m->health += heal_power;
}
```

Ввод вывод в C++. iostream

iostream

```cpp
#include <iostream>

int main
{
    std::cout << "Hello world! " << 42 << std::endl;
    int a;
    std::cin >> a;
}
```

iostream

```
#include <iostream>
using namespace std;

int main
{
    cout << "Hello world! " << 42 << endl;
    int a;
    cin >> a;
}
```

Раздельная компиляция

Компилируемый язык

Язык C является компилируемым языком программирования
Компиляция – преобразование текста программы(исходного кода) в машинный код.

Зачем разбивать программу на файлы?

1. С небольшими файлами удобнее работать
2. Ускорения повторной компиляции при небольших изменениях
3. Структуирование кода

Общая схема сборки программы