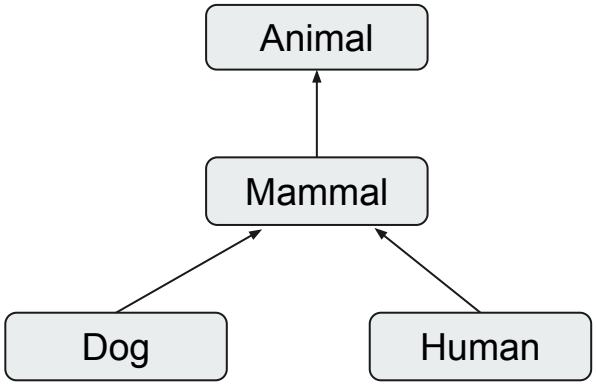
Наследяване++

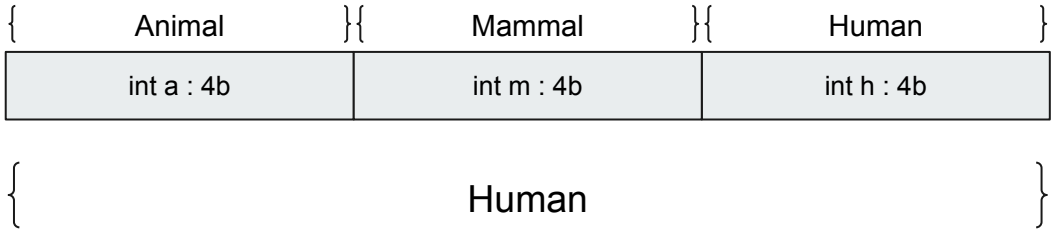
Наследяване

- Наследяването на базов клас извън контекста на полиморфизъм:




```
class Animal {int a;};
class Mammal : Animal {int m;};
class Dog : Mammal {int d;};
class Human : Mammal {int h;};
```

- Разположение в паметта:



Видове наследяване

- Какво е наследяването по подразбиране при класовете и структурите?
- Кога public?
- Къде private?
- Защо protected?

Въпрос:

- Как бихте нарисували структурата?

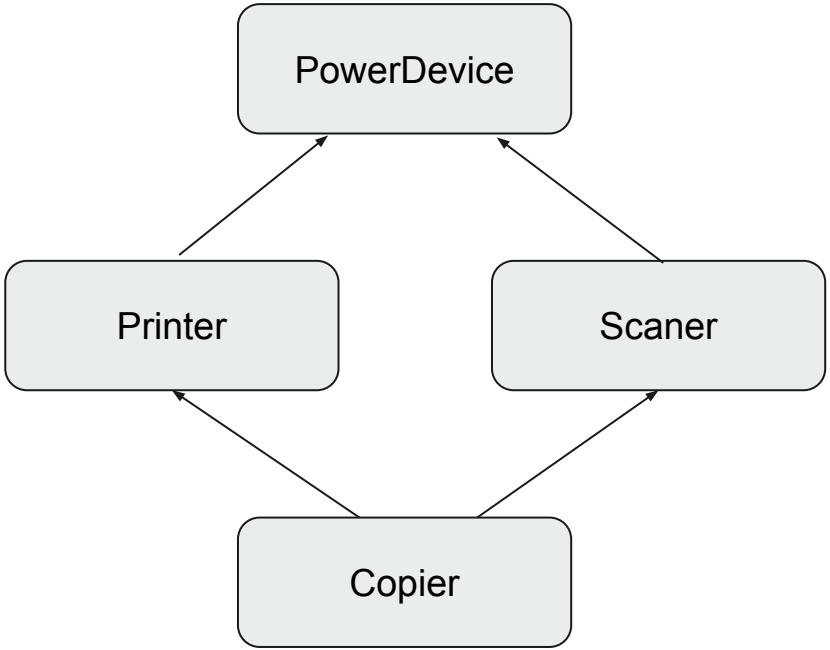
```
struct A {};

struct B : A {};

struct C : private B {};

struct D : C, B {};

struct E : A, C, private B {};
```

Въпрос:

- Има ли проблем в структурата E?
- Как можем да го решим?

Multiple inheritance

- Диамантен проблем



Multiple inheritance

- Диамантен проблем?
- Диамантено решение?

Multiple inheritance

- Диамантен проблем?
- Диамантено решение?
- Ммм не? … Пак virtual

Multiple inheritance

- static_cast

Проблем:

Как да постигнем различно поведение (функционалност) за обекти с общ родител, които “overload”-ват дадена функция?

Пойнтъри към функции

```
//typedef void(*Func)();

using Func = void (*)();

void bark() { std::cout << “bau” << std::endl; }
void moo() { std::cout << “moo” << std::endl; }

void (*)() get_animal_voice(const std::string& animal) {
    if (animal == “dog”) {
        return bark;
    }
   if (animal == “cow”)  {
       return moo;
   }
   return [] { std::cout << “error” << std::endl; };
}

int main(int argc, char **argv) {
     Func f = get_animal_voice(argv[1]);
     f();
}
```

Структури с пойнтъри към функции

```
using SpeakFunc = void (*)(Animal *);
using GetOldFunc = void (*)(Animal *, int);

struct Animal {
    SpeakFunc speak;
    GetOldFunc get_older;
    const char *name;
    int age;
};
```

Структури с пойнтъри към функции

```
void bark(Animal *animal) { std::cout << animal->name << “says bau” << std::endl; }
void dog_aging(Animal *animal, int i) { animal->age += i * 7; }

Animal GetDog(const char *name) {

        Animal dog = Animal{
                bark,
                dog_aging,
                name,
                0 /*age */
        };
        return dog;
}
```

Структури с пойнтъри към функции

```
Animal GetCat(const char *name) {

        Animal cat = Animal{
                miau,
                cat_aging,
                name,
                0 /*age */
        };
        return cat;
}
```

Структури с пойнтъри към функции

```
int main() {

    Animal dog = GetDog(“6aro”);
    Animal cat = GetCat(“pisana”);

    dog.speak(&dog);
    cat.age(&cat, 3);
}
```

Проблем:

Как да постигнем различно поведение (фунционалност) за обекти с общ родител, които “overload”-ват дадена функция?

```
class Shape {
      public:
            int calculateArea() { return 0; }
};

class Rect : public Shape {
            float a, b;
      public:
            float calculateArea() { return a*b; }
};

class Circle : public Shape {
            float rad;
      public:
            float calculateArea() { return 3.14*rad*rad; }
};
```

Проблем:

```
int main() {

   Circle circle;
   Rectangle rectangle;

   circle.radius = 5.0;
   rectangle.width = 4.0;
   rectangle.height = 6.0;

   Shape* shapes[] = { (Shape*)&circle, (Shape*)&rectangle };

  for (int i = 0; i < 2; i++) {
      double area = shapes[i]->calculateArea();
      printf("Area: %.2f\n", area);
   }

 return 0;
}
```

Решение:

- Пойнтъри към функции? (Function pointers): https://godbolt.org/z/7sfjrG4no
- C-стил решение

Решение в C++ стил:

https://godbolt.org/z/o9zo3qsxG

Въпрос:

class A {};
Каква е стойността на sizeof(A)?

class B { int foo() {} };
Каква е стойността на sizeof(B)?

class C { virtual void foo() {} };
Каква е стойността на sizeof(C)?

Solution: https://godbolt.org/z/5731KTnsx

Виртуални таблици

- Структури с указатели към функции които могат да се подменят
- Рънтайм полиморфизмът в C++ се изпълнява посредством виртуални таблици и виртуални пойнтъри а.к.а vptr.

Virtual:

- Виртуални функции.
- “Чисто” виртуални (pure virtual) функции
- Прилики/разлики

Абстрактен клас:

- Няма ключова дума abstract в C++
- Какво е абстрактен клас?
- Как тогава да създадем абстрактен клас?

Абстрактен клас:

- Дефиниране на поведение/интерфейс.

Brainfuck интерпретатор за 15 мин

https://godbolt.org/z/sjMsbYPsd

Помислете какъв дизайн можем да направим, за да направим ООП ориентиран интерпретатор?