

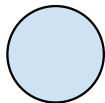
Наследяване и виртуални функции. Въведение

Калин Георгиев

22 април 2020 г.

Моделиране на различни обекти с общи свойства

Еднакви и различни



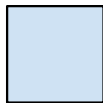
Circle

Дефиниран чрез:

- радиус
- координати на центъра

Свойства:

- периметър
- лице



Square

Дефиниран чрез:

- страна
- координати на центъра
- ъгъл с оста

Свойства:

- периметър
- лице



Isosceles

Дефиниран чрез:

- бедро
- основа
- координати на медицентъра
- ъгъл с оста

Свойства:

- периметър
- лице



Polygon

Дефиниран чрез:

- списък координати на върхове

Свойства:

- периметър
- лице
- брой върхове
- дали е изпъкнал

Множество от различни обекти

```
int main ()
{
    Square* squares[] =
    {new Square (2,0,0,0),
      new Square (4,0,0,0),
      new Square (3,0,0,0)};

    Circle* circles[] =
    {new Circle (2,0,0),
      new Circle (4,0,0)};

    cout << sumSurf<Square> (squares,3) +
           sumSurf<Circle> (circles,2);

    return 0;
}
```

```
template <typename F>
double sumSurf (F* figures[], int n)
{
    double sum = 0;

    for (int i = 0; i < n; i++)
        sum += figures[i]->surface();

    return sum;
}
```

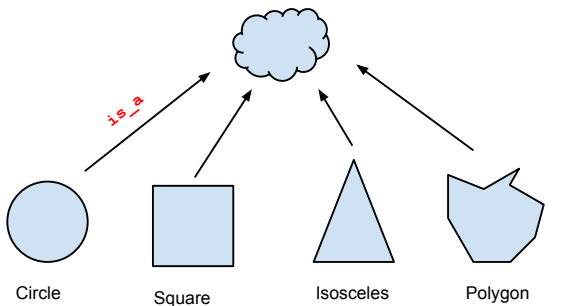
"Абстрахиране" от конкретния тип

Какво е фигура? Йерархия от фигури

Figure: абстрактно понятие

Свойства:

- периметър
- лице



Дефиниран чрез:
 - радиус
 - координати на центъра
 Свойства:

Дефиниран чрез:
 - страна
 - координати на центъра
 - ъгъл с оста
 Свойства:

Дефиниран чрез:
 - бедро
 - основа
 - координати на медицентъра
 - ъгъл с оста
 Свойства:

Дефиниран чрез:
 - списък координати на върхове
 Свойства:
 - брой върхове
 - дали е изпъкнал

Полиморфизъм

```
int main ()
{
    Figure* figures[] =
        {new Square (2,0,0,0),
          new Circle (2,0,0),
          new Square (4,0,0,0),
          new Square (3,0,0,0),
          new Circle (4,0,0)};

    cout << sumSurf (figures,5);

    return 0;
}
```

```
template <typename F>
double sumSurf (F* figures[], int n)
{
    double sum = 0;

    for (int i = 0; i < n; i++)
        sum += figures[i]->surface();

    return sum;
}
```

Клас Figure

```
class Figure
{
    public:
        char label[15];
        double surface () {
            assert (false);
            return -1;
        }
};
```

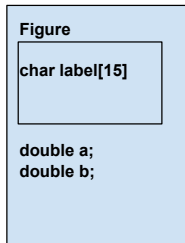
Figure

char label[15]

Наследяване на клас Figure

```
class Rectangle : public Figure
{
    public:
        double a,b;
        double surface ()
        {return a*b;}
};
```

Rectangle



Съвместимост на Figure и Rectangle

```
Rectangle r1 (2,4);  
Figure f = r1;  
cout << f.surface();
```

Figure

char label[15]

Rectangle

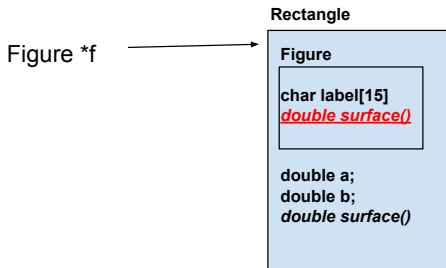
Figure

char label[15]

double a;
double b;

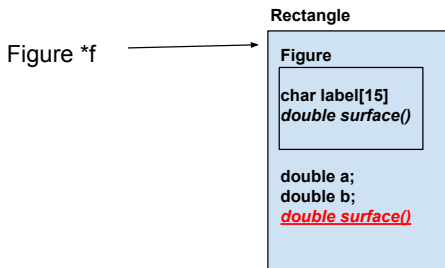
Съвместимост на указатели към Figure и Rectangle

```
Rectangle r1 (2,4);  
Figure *f = &r1;  
cout << f->surface();
```

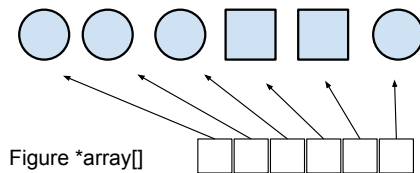


Ако Figure::surface е виртуален

```
cout << f->surface();
```



Масив от указатели към фигури



Благодаря ви за вниманието!