«ԾՐԱԳՐԱՎՈՐՄԱՆ ՀԻՄՈՒՆՔՆԵՐ» դասընթաց

այլ ոլորտներից դեպի տեխնոլոգիական ոլորտ սկսնակների համար

edu2020.am

ԴԱՍ #16

ՀԱՅ-ՌՈՒՍԱԿԱՆ ՀԱՄԱԼՍԱՐԱՆ

ՀԱՅԱՍՏԱՆԻ ՀԱՆՐԱՊԵՏՈՒԹՅԱՆ
ԲԱՐՁՐ ՏԵԽՆՈԼՈԳԻԱԿԱՆ
ԱՐԴՅՈՒՆԱԲԵՐՈՒԹՅԱՆ ՆԱԽԱՐԱՐՈՒԹՅՈՒՆ

Բազմաձևություն

- Բազմաձևությունը (polymorphism) բնութագրվում է հետևյալ արտահայտությամբ - «մեկ ինտերֆեյս, շատ մեթոդներ»

- Բազմաձևությունը օբյեկտա-կողմնորոշված ծրագրավորման երրորդ սկզբունքն է

Ժառանգում (inheritance)

```
class Base {
 public:
  std::string getName() {
    return "Base class function\n";
  }
};

class Derived: public Base {
 public:
  std::string getName() {
    return "Derived class function\n";
  }
};

int main() {
  Derived derived;
  Base* basePointer = &derived;
  std::cout << basePointer->getName();
}
```



https://repl.it/@HaykAslanyan/polymorphism1

Ժառանգում (inheritance)

```
class Base {
 public:
  std::string getName() {
    return "Base class function\n";
  }
};

class Derived: public Base {
 public:
  std::string getName() {
    return "Derived class function\n";
  }
};

int main() {
  Derived derived;
  Base* basePointer = &derived;
  std::cout << basePointer->getName();
}
```

Կկանչվի բազային դասի ֆունկցիան

https://repl.it/@HaykAslanyan/polymorphism1

Բազմաձևություն (օրինակ 1)

```
class Base {
 public:
  virtual std::string getName() {
    return "Base class function\n";
  }
};

class Derived: public Base {
 public:
  virtual std::string getName() {
    return "Derived class function\n";
  }
};

int main() {
  Derived derived;
  Base* basePointer = &derived;
  std::cout << basePointer->getName();
}
```

Ավելացվել է virtual բանալի բառը

Կկանչվի ժառանգված դասի ֆունկցիան

https://repl.it/@HaykAslanyan/polymorphism2

Բազմաձևություն (օրինակ 2)

```
class Base {
 public:
  virtual std::string getName() {
    return "Base class function\n";
  }
};
class Derived1: public Base {
 public:
  virtual std::string getName() {
    return "Derived 1 class function\n";
  }
};

class Derived2: public Derived1 {};

class Derived3: public Derived2 {
 public:
  virtual std::string getName() {
    return "Derived 3 class function\n";
  }
};
```

Derived2-ը կժառանգի Derived1::getName() ֆունկցիան

https://repl.it/@HaykAslanyan/polymorphism3

Բազմաձևություն (օրինակ 2)

```
class Base {
 public:
  virtual std::string getName() {
    return "Base class function\n";
  }
};
class Derived1: public Base {
 public:
  virtual std::string getName() {
    return "Derived 1 class function\n";
  }
};

class Derived2: public Derived1 {};

class Derived3: public Derived2 {
 public:
  virtual std::string getName() {
    return "Derived 3 class function\n";
  }
};
```

```
void print(Base** baseDP, int size) {
  for (int i = 0; i < size; i++) {
    std::cout << baseDP[i]->getName();
  }
}

int main() {
  Derived1 derived1_1;
  Derived2 derived2_1, derived2_2;
  Derived3 derived3_1, derived3_2;

  const int size = 5;
  Base* basePointerArray[size];
  basePointerArray[0] = &derived2_1;
  basePointerArray[1] = &derived2_2;
  basePointerArray[2] = &derived3_1;
  basePointerArray[3] = &derived3_2;
  basePointerArray[4] = &derived1_1;
  print(basePointerArray, size);
}
```

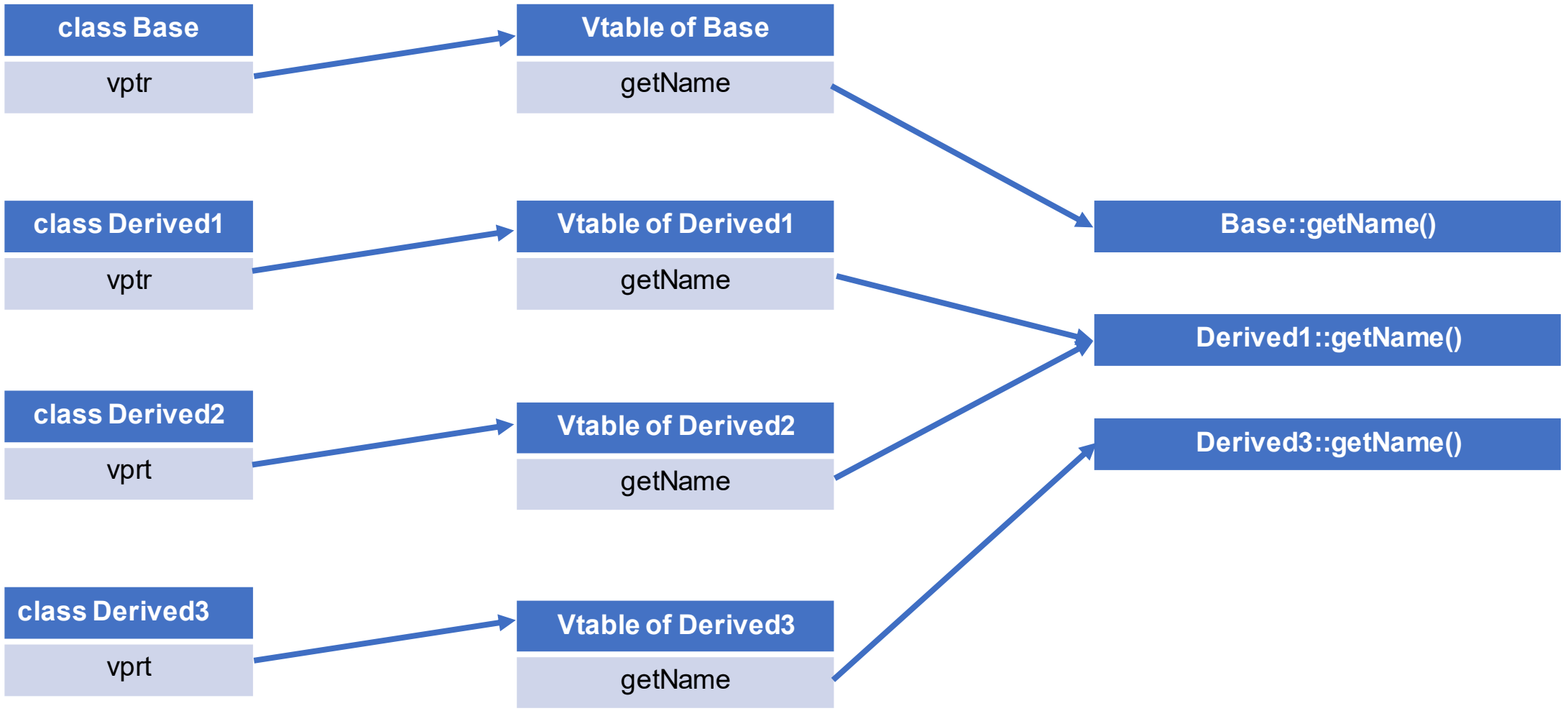
Կկանչվի իրական օբյեկտների getName ֆունկցիան

https://repl.it/@HaykAslanyan/polymorphism3

Բազմաձևություն

- Վիրտուալ ֆունկցիաների կանչի մշակումը ավելի ժամանակատար է, քան սովորական ֆունկցիաների
- Թարգմանիչը հավելյալ ցուցիչ (*vptr*) է հատկացնում յուրաքանչյուր օբյեկտի համար, որի համապատասխան դասն ունի մեկ կամ մի քանի վիրտուալ ֆունկցիաներ
- Նաև, թարգմանիչը յուրաքանչյուր դասի համար ստեղծում է *vtable*, որը պարունակում է ցուցիչներ ֆունկցիաների վրա

Վիրտուալ ֆունկցիաների կանչ



Վիրտուալ ֆունկցիաների կանչ

- Յուրաքանչյուր օբյեկտի կառուցիչում թարգմանիչը ավտոմատ ստեղծում է *vptr* ցուցիչը, որը ցույց է տալիս համապատասխան դասի *vtable*-ի վրա
- `baseDP[i]->getName()` կանչի ժամանակ իրական օբյեկտի *vptr*-ի միջոցով դիտարկվում է *vtable*-ը և կանչվում այդտեղի getName() ֆունկցիան
- Քանի որ Derived2-ը ժառանգում է Derived1::getName(), ապա `baseDP[0]`-ի դեպքում կկանչվի Derived1::getName()
- Իրական օբյեկտի տիպի որոշումը կատարվում է ծրագրի աշխատանքի ժամանակ (run time)

Փլուզիչ (destructor)

```
#include<iostream>
class base {
  public:
    base() {
      std::cout << "Constructing base \n";
    }
    ~base() {
      std::cout << "Destructing base \n";
    }
};
class derived : public base {
  public:
    derived() {
      std::cout << "Constructing derived \n";
    }
    ~derived() {
      std::cout << "Destructing derived \n";
    }
};
```

```
int main() {
  base *pb = new derived();
  delete pb;
}
```

Կկանչվի բազային դասի փլուզիչը, իսկ ժառանգված դասինը՝ ոչ

https://repl.it/@HaykAslanyan/notvirtualdestructor

Վիրտուալ փլուզիչ (virtual destructor)

```
#include<iostream>
class base {
  public:
    base() {
      std::cout << "Constructing base \n";
    }
    virtual ~base() {
      std::cout << "Destructing base \n";
    }
};
class derived : public base {
  public:
    derived() {
      std::cout << "Constructing derived \n";
    }
    virtual ~derived() {
      std::cout << "Destructing derived \n";
    }
};
```

```
int main() {
  base *pb = new derived();
  delete pb;
}
```

Կկանչվի ժառանգված և բազային դասերի փլուզիչները

Ավելացվել է virtual բանալի բառը

https://repl.it/@HaykAslanyan/virtualdestructor

Աբստրակտ դասեր

- Երբեմն հնարավոր չէ բազային դասի բոլոր ֆունկցիաները իրականացնել
- Օրինակ դիտարկենք Shape դասը
- `virtual double getArea() = 0;` ֆունկցիան մաքուր վիրտուալ ֆունկցիա է, այն չունի իրականացում
- Այն դասը, որը պարունակում է մաքուր վիրտուալ ֆունկցիա կոչվում է աբստրակտ
- Աբստրակտ դասի օբյեկտ չի թույլատրվում սարքել, սակայն կարող ենք ունենալ այդ տիպի ցուցիչ
- Աբստրակտ դասից կարող ենք ժառանգել այլ դասեր

```
class Shape {
 public:
  virtual ~Shape() {}
  virtual double getArea() = 0;
  void printArea() {
    std::cout << getArea() << "\n";
  }
};
```



https://repl.it/@HaykAslanyan/abstractclass

Աբստրակտ դասեր

Ժառանգում Shape աբստրակտ դասից և getArea() ֆունկցիայի վերասահմանում

```
class Square : public Shape {
 public:
  Square(double s) : side(s) {}
  virtual ~Square() {}

  virtual double getArea() {
    return side * side;
  }
 private:
  double side;
};
```

```
class Triangle : public Shape {
 public:
  Triangle(double s1, double s2, double s3) :
    side1(s1), side2(s2), side3(s3) {}
  virtual ~Triangle() {}
  virtual double getArea() {
    double p = (side1 + side2 + side3) / 2;
    return sqrt(p * (p - side1) * (p - side2) *
(p - side3));
  }
 private:
  double side1;
  double side2;
  double side3;
};
```

https://repl.it/@HaykAslanyan/abstractclass

Աբստրակտ դասեր

```
int main() {
  const int size = 5;
  Shape* shapePointerArray[size];
  // shapePointerArray[0] = new Shape();
  shapePointerArray[0] = new Square(1.1);
  shapePointerArray[1] = new Square(2.2);
  shapePointerArray[2] = new Triangle(3, 4, 5);
  shapePointerArray[3] = new Triangle(30, 40, 50);
  shapePointerArray[4] = new Triangle(300, 400, 500);

  for (int i = 0; i < 5; i++) {
    shapePointerArray[i]->printArea();
  }

  for (int i = 0; i < 5; i++) {
    delete shapePointerArray[i];
  }
}
```

Չի կարելի ստեղծել աբստրակտ դասի օբյեկտ

Կտպվի իրական օբյեկտների մակերեսները



https://repl.it/@HaykAslanyan/abstractclass

Խնդիրներ

- Գրել worker դասը և ժառանգել teacher, musician, doctor դասերը
- Գրել fruit դասը և ժառանգել apple, banana, orange դասերը
- Գրել animal դասը և ժառանգել dog, cat, lion դասերը

Դասի ստատիկ անդամներ և ֆունկցիաներ

- Դասի ստատիկ անդամները և ֆունկցիաները ընդհանուր են բոլոր օբյեկտների համար
- Դասի ստատիկ անդամները կարող են օգտագործվել դասի բոլոր ֆունկցիաներում
- Դասի ստատիկ ֆունկցիաները կարող են օգտագործել միայն դասի ստատիկ անդամներին
- Դասի ստատիկ անդամներն ու ֆունկցիաները հնարավոր է օգտագործել նույնիսկ, եթե չկան այդ դասի օբյեկտներ
- Ստատիկ անդամները և ֆունկցիաները ժառանգված դասում հասանելի են նույն սկզբունքով, ինչպես այլ անդամները և ֆունկցիաները

Դասի ստատիկ անդամներ և ֆունկցիաներ

```
class person {
 protected:
    std::string name;
    int age;
    static unsigned objectsCount;
 public:
    person(std::string n, int a) : name(n), age(a) {
      objectsCount++;
    }
    ~person() {
      objectsCount--;
    }
    void printInfo() {
      std::cout << this->name << " "
                << this->age << "\n";
    }
    static unsigned getObjectsCount() {
      return objectsCount;
    }
};
unsigned person::objectsCount = 0;
```

Ընդհանուր է դասի բոլոր օբյեկտների համար

Ստատիկ ֆունկցիաները կարող են օգտագործել միայն ստատիկ անդամներ

Ստատիկ անդամի սկզբնարժեքավորում

https://repl.it/@HaykAslanyan/staticmembers

Դասի ստատիկ անդամներ և ֆունկցիաներ

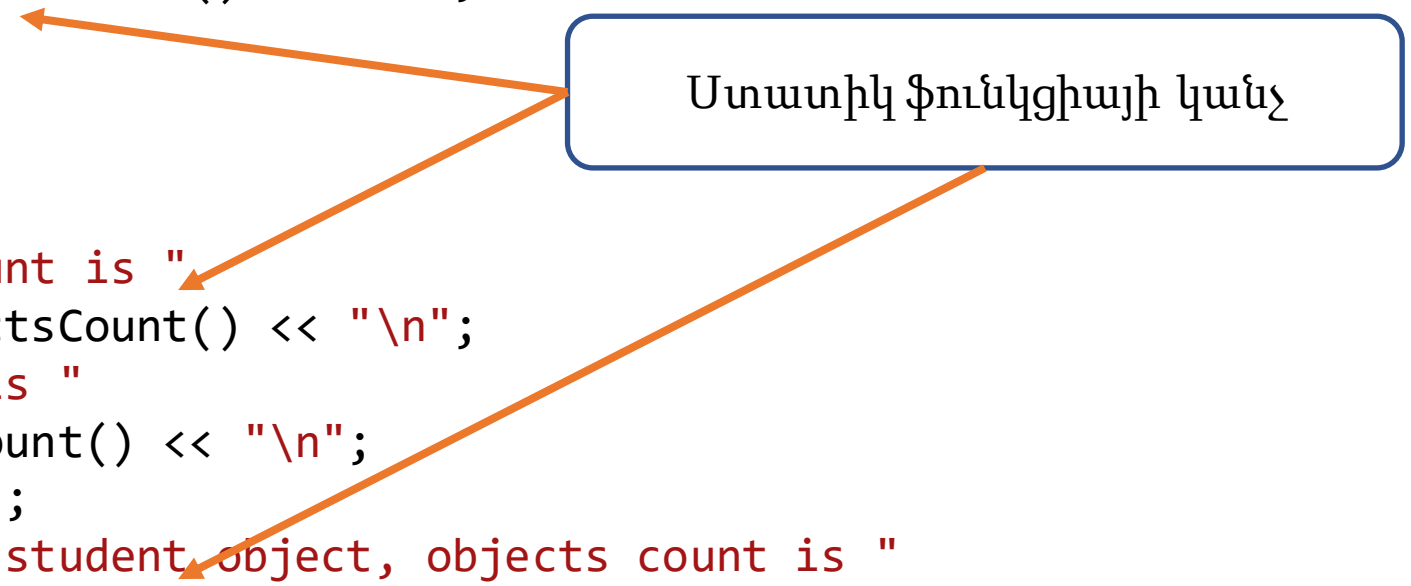
Ժառանգվում են նաև `objectsCount` անդամը և `getObjectsCount()` ֆունկցիան

```
class student : public person {
  protected:
    unsigned year;
    float gpa;
 public:
    student(std::string n, int a, unsigned y, float g) :
      person(n, a), year(y), gpa(g) {}

};
```

https://repl.it/@HaykAslanyan/staticmembers

Դասի ստատիկ անդամներ և ֆունկցիաներ

```
int main() {
  std::cout << "At first, objects count is "
            << person::getObjectsCount() << "\n";
  person p("Bob", 25);
  p.printInfo();

  person p2("Nairi", 38);
  std::cout << "Now objects count is "
            << person::getObjectsCount() << "\n";
  std::cout << "Objects count is "
            <<  p2.getObjectsCount() << "\n";
  student s("Artyom", 38, 4, 4);
  std::cout << "After creating student object, objects count is "
            << person::getObjectsCount() << "\n";
}
```



https://repl.it/@HaykAslanyan/staticmembers

this ցուցիչ

- Դասի յուրաքանչյուր օբյեկտ ունի հասանելիություն իր հասցեին *this* ցուցիչի միջոցով

- *this* ցուցիչը հասանելի է դասի բոլոր ֆունկցիաներում
  - բացառություն են կազմում դասի static ֆունկցիաները

this ցուցիչ

```
class person {
 private:
    std::string name;
    int age;

 public:
    person(std::string n, int a) :
           name(n), age(a) {}
    void printInfo() {
      std::cout << this->name << " "
                << this->age << "\n";
    }
    void changeName(std::string name) {
      this->name = name;
    }
};
```

```
int main() {
  person p("Bob", 25);
  p.printInfo();

  p.changeName("Alex");
  p.printInfo();
}
```

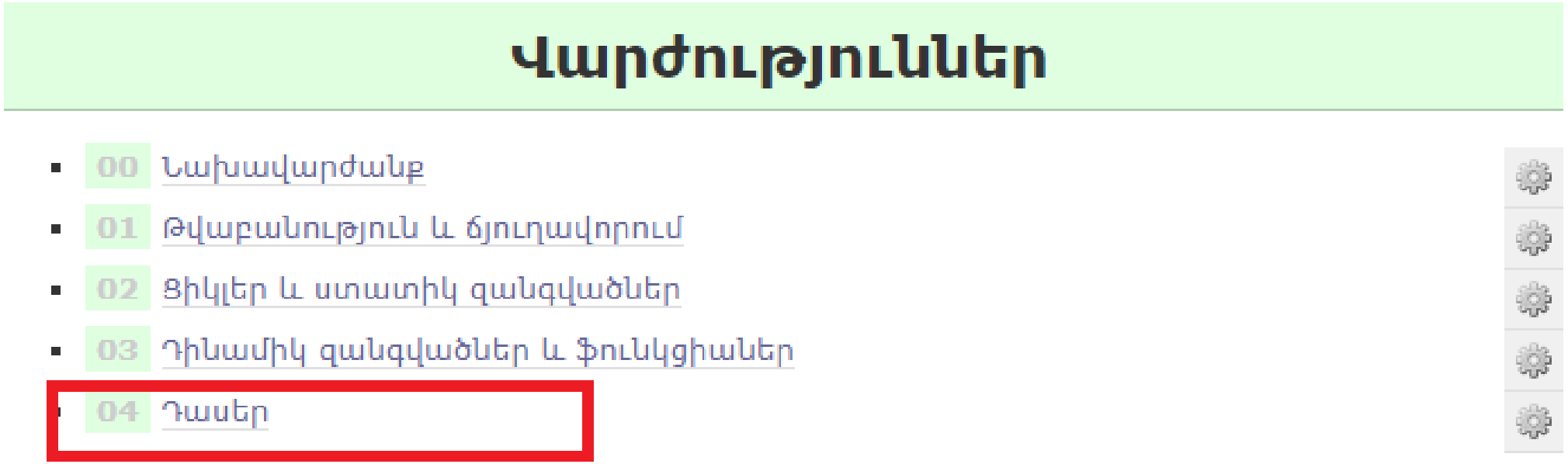
Օբյեկտի name անդամ

changeName ֆունկցիայի name արգումենտ

https://repl.it/@HaykAslanyan/thispointer

Տնային աշխատանք

Տնային աշխատանք 15, 16



Ամփոփում

- c++ լեզվի դերը
- Ծրագրի կատարման պրոցեսը
- Տիպերը c++-ում
- Տվյալների ներմուծում և արտածում
- Թվաբանական գործողություններ
- Համեմատման օպերատորներ
- Ճյուղավորման օպերատորներ
- Ցիկլեր
- Ցուցիչներ, հղումներ
- Ստատիկ և դինամիկ զանգվածներ
- Ֆունկցիաներ
- Ռեկուրսիվ ֆունկցիաներ
- Զանգվածների տրտավորման 3 ալգորիթմներ
- Դասեր
- Դասերի ժառանգականություն
- Դասերի բազմաձևություն

Ընտրհավկալություն. Հարցե՞ր