Класове. Дефиниране. Обекти. Конструктори. Указатели към обекти на класове. Масиви и обекти.

гл.ас. д-р. Нора Ангелова

КОНТРОЛНА ЛЕКЦИИ 1

Дадена е структура от данни Book с две полета - name (до 30 символа) и author (указател към тип char).

Дадена е структура от данни Student с две полета - fn (цяло число) и указател към книга.

- **Дефинирайте 2-те структури от данни**
- **Дефинирайте масив от указатели към студенти - list с максимален размер 30.**
- **Инициализирайте първия елемент на масива**
- **Изведете на екрана името на автора на книгата, която е притежание на първия студент.**

РЕШЕНИЕ

```
struct Book {
  char name[30];
  char* author;
};

struct Student {
  int fn;
  Book* studentBook;
};
```

РЕШЕНИЕ

```
Book bk;
cin.getline(bk.name, 30);
bk.author = "Author Name";

Student st;
st.fn = 12345;
st.studentBook = &bk;

Student* list[30] = {&st};
cout << list[0]->studentBook->author;
```

КЛАСОВЕ

В езика C++ има стандартен набор от типове данни като int, double, float, char, string и др. Този набор може да бъде разширен чрез дефинирането на класове.

Дефинирането на клас въвежда нов тип, който може да бъде интегриран в езика.

Класовете са в основата на обектно-ориетираното програмиране, за което е предназначен езика C++.

КЛАСОВЕ

Може ли да използваме структура rat като тип данни рационално число?

КЛАСОВЕ

- Подобни на структурите
- С допълнителни ограничение по отношение на правата за достъп

Напомняне - абстракция

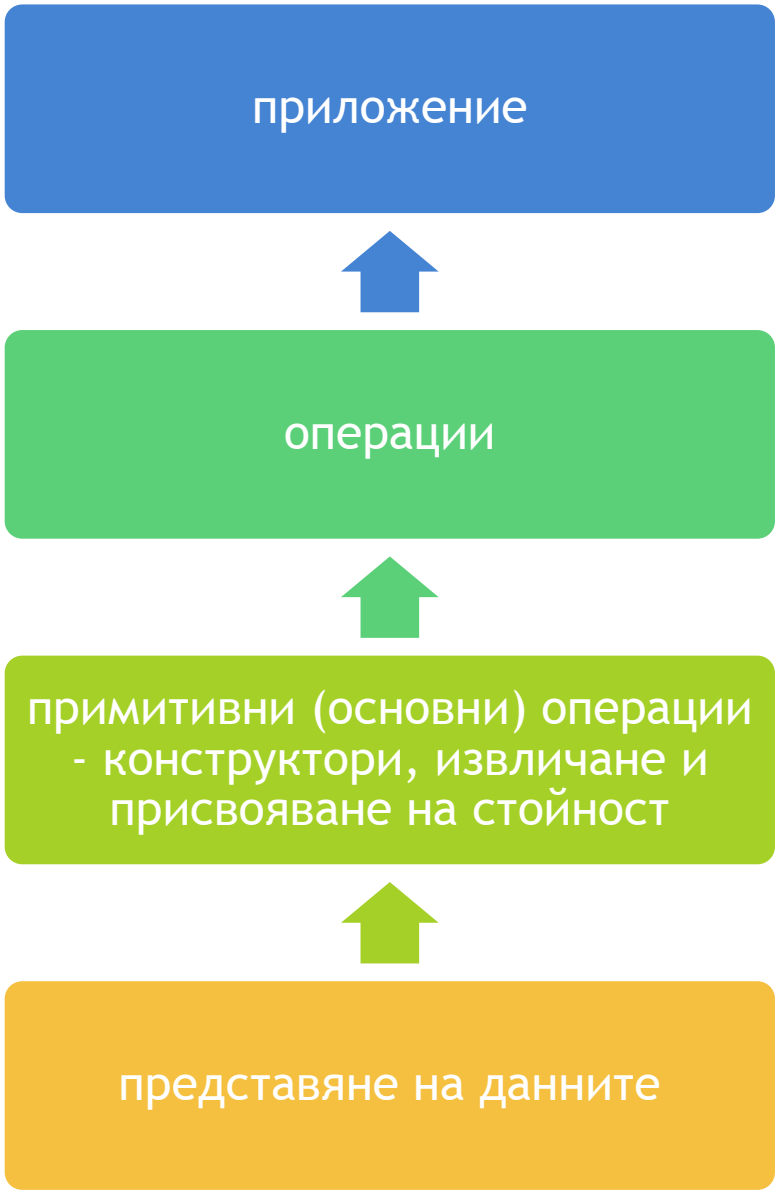
Идея: методите за използването на данните се разделят от тяхното представяне.

1. Всяка програма се проектира така, че да работи с „абстрактни данни“ – данни с неясно представяне.
2. Представянето на данните се конкретизира с помощта на множество функции – конструктури, селектори (гетъри), мутатори (сетъри), предикати.

НАПОМНЯНЕ

Абстрактен тип данни – тип данни, за който се изисква скриване на реализацията на типа и неговото „поведение“ се дефинира от множество от данни и множество от операции.

КЛАСОВЕ


приложение
операции
примитивни (основни) операции - конструктори, извличане и присвояване на стойност
представяне на данните

КЛАСОВЕ

- Всяко ниво използва единствено средствата на предходното

Какви са предимствата ?

КЛАСОВЕ

```
struct rat {
  int numer;
  int denom;
};
```

две полета:

numer – числителя

denom – знаменателя

двете полета се наричат **член-данни на структурата**

КЛАСОВЕ

- Конструктори

Конструкторите са член-функции, чрез които се инициализират променливите на структурата.

* Те винаги имат за име името на структурата.
* Не се указва тип на връщания резултат.

КЛАСОВЕ

- Конструктори

Конструктор без параметри

```
struct rat {
   ...
   rat();
};
```

Нарича се

конструктор по подразбиране

КЛАСОВЕ

- Конструктори

```
rat::rat() {
   numer = 0;
   denom = 1;
}


rat number;

// number се инициализира с
// рационалното число 0/1
```

КЛАСОВЕ

- Конструктори

Конструктор с параметри

```
struct rat {
   ...
   rat(int, int);
};
```

КЛАСОВЕ

- Конструктори

```
rat::rat(int x, int y) {
  numer = x;
  denom = y;
}

rat number(3,4);
// number се инициализира с
// рационалното число 3/4

rat number2(4,5);
// number2 се инициализира с
// рационалното число 4/5
```

КЛАСОВЕ

- Мутатори (сетъри)

Член-функции, които променят член-данните на структурата.

```
struct rat {
  ...
  void setNumer(int);
};
```

```
void rat::setNumer(int x) {
  numer = x;
}
```

КЛАСОВЕ

- Функции за достъп (гетъри)

Член-функции, които извличат информация за член-данните на структурата.

Тези функции не променят член-данните на структурата.

```
struct rat {
  ...
  int getNumer() const;
};

int rat::getNumer() const {
  return numer;
}
```

КЛАСОВЕ

- Операции

```
struct rat {
  ...
};

rat multRats(rat const & r1, rat const & r2) {
  rat r(
    r1.getNumer() * r2.getNumer(),
    r1.getDenom() * r2.getDenom());
  return r;
}
```

КЛАСОВЕ

- Спецификатори за достъп

```
struct rat {
 private:
   int numer;
   int denom;
 public:
   rat();
   rat(int, int);

   int getNumer() const;
   int getDenom() const;
};
```

КЛАСОВЕ

- Дефиниране на клас

```
class rat {
 private:
   int numer;
   int denom;
 public:
   rat();
   rat(int, int);

   int getNumer() const;
   int getDenom() const;
};
```

КЛАСОВЕ

- По подразбиране членовете на структура имат public достъп
- По подразбиране членовете на класовете имат private достъп

КЛАСОВЕ

- Обекти

Екземпляри на класа/структурата

<дефиниция-на_обект_на_клас> ::= <име_на_клас> <обект>
[=<име_на_клас>(<фактически_параметри>)]опц
{,<обект>[=<име_на_клас>(<фактически_параметри>)]опц
}орс
{, <обект>(<фактически_параметри>)}опц
{, <обект> = <вече_дефиниран_обект>}опц;
<обект> ::= <идентификатор>

Пример:

rat chislo, chislo2=rat(1,7), chislo3(-3,4);

са обекти инициализирани съответно с рационалните числа: 0/1, 1/7, -3/4

КЛАСОВЕ

- Капсулиране на информация

Спецификаторът private, забранява използването на член-данните numer и denom извън класа.

Процесът на скриване на информация се нарича капсулиране на информация.

- Интерфейс

Член-функциите на класа rat са обявени като public. Те са видими извън класа и могат да се използват от външни функции.

Те се наричат интерфейс на класа.

Дефиниране на клас

- Декларация на клас
- Дефиниция на неговите член-функции

Дефиниране на клас

- Декларация на клас
- Дефиниция на неговите член-функции

* Възможно е член-функциите на класа да се дефинират в тялото на класа.

```
class Test {
  int testFunc () {
    …
  }
};
```

В този случай те се <u>третират</u> като вградени.

ВГРАДЕНИ ФУНКЦИИ

- Кодът на тези функции не се съхранява на едно място, а се копира на всяко място в паметта, където има обръщение към тях.

- Използват се като останалите функции, но заглавието им се предшества от модификатора inline(при декларация и дефиниция).

Пример:

```
inline int func(int a, int b) {
  return (a+b)*(a-b);
}
```

Област на клас

- Деклариран глобално - започва от декларацията и продължава до края на програмата.
- Деклариран локално (вътре във функция или в тялото на клас) - всички негови член-функции трябва да са вградени (inline). В противен случай ще се получат функции, дефинирани във функция, което не е възможно.

КЛАСОВЕ

- Дефиницията на клас не заделя памет за него. Памет се заделя едва при дефинирането на обект от класа.
- Достъпът до компонентите на обектите (ако е възможен) се осъществява чрез задаване на името на обекта и името на данната или метода, разделени с точка.
  Изключение от това правило правят конструкторите

КЛАСОВЕ

- Кодът на методите на класа не се копира във всеки обект, а се намира само на едно място в паметта.

- По какъв начин методите на един клас “разбират” за кой обект на този клас са били извикани?

Указател this

Компилаторът

- Преобразува всяка член-функция на даден клас в обикновена функция с уникално име и един допълнителен параметър – указателят this.

int rat::getNumer() {…} -> int rat::getNumer(rat* this) {…}

- Всяко обръщение към член-функция се транслира в съответствие първата част

rat n;

n.getNumer(); -> getNumer(&n);

Указател this

Да направим функцията multRats член-функция на класа

Външна функция:

```
rat multRats(rat const & r1, rat const & r2) {
  rat r(
    r1.getNumer() * r2.getNumer(),
    r1.getDenom() * r2.getDenom());
  return r;
}
```

Вътрешна функция:

```
void rat::multRats(rat const & r1, rat const & r2) {
  numer = r1.numer * r2.numer;
  denom = r1.denom * r2.denom;
}
```

КРАЙ