Класове

Какво са класовете?

- Основен инструмент на ООП
- Средство за дефиниране на абстрактни типове данни
- Синтактична конструкция, която позволява логическо групиране на данни и операциите над тях

Дефиниция на клас

- Декларации на **член-данни (полета)**
- Декларации на **член-функции (методи)**
  - конструктори
  - селектори
  - мутатори
  - деструктор

Дефиниция на клас

- <клас> ::= class <име_на_клас> { <тяло> };
- <име_на_клас> често е съществително име с главна буква
- <тяло> ::= { <декларация>; }
- <декларация> ::= <член-данна> | <конструктор> | <мутатор> | <селектор> | <деструктор>
- <член-данна> ::= [<достъп>:] <тип> <име> {, <име>}
- <конструктор> ::= [<достъп>:] <име_на_клас>(<параметри>)
- <мутатор> ::= [<достъп>:] <тип> <име>(<параметри>)
- <селектор> ::= [<достъп>:] <тип> <име> (<параметри>) const
- <деструктор> ::= [<достъп>:] ~<име_на_клас>()
- <достъп> ::= private | protected | public

Дефиниция на клас

- Може да присъства само един път в даден файл
- Обикновено се пише в заглавен (header) файл с разширение .h
- Файловете, които използват класа, включват дефиницията му чрез включване на заглавния файл с #include

```
class Rational {
private:
  int numer, denom;
public:
  Rational();
  int getNumerator() const;
  void read();
};
```

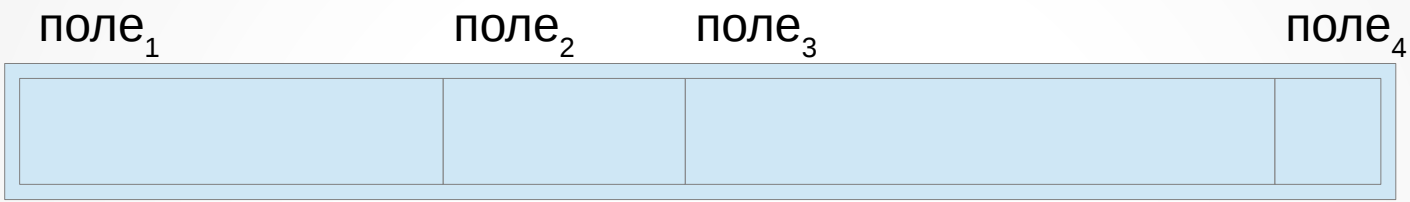
Дефиниция на клас

- Конструкторите и деструкторите нямат тип!
- Деструкторът няма параметри!
- Прието е член-данни и член-функции да са разделени
- Директната рекурсията е забранена, както при записи ~~class Employee { Employee boss; ... };~~
- Индиректната рекурсия (чрез указател) е позволена
  class Employee { Employee* boss; ... };
- Член-функциите могат да са от всякакъв тип, включително и същия клас:
- class Employee { ... Employee getBoss() const; };

Обекти

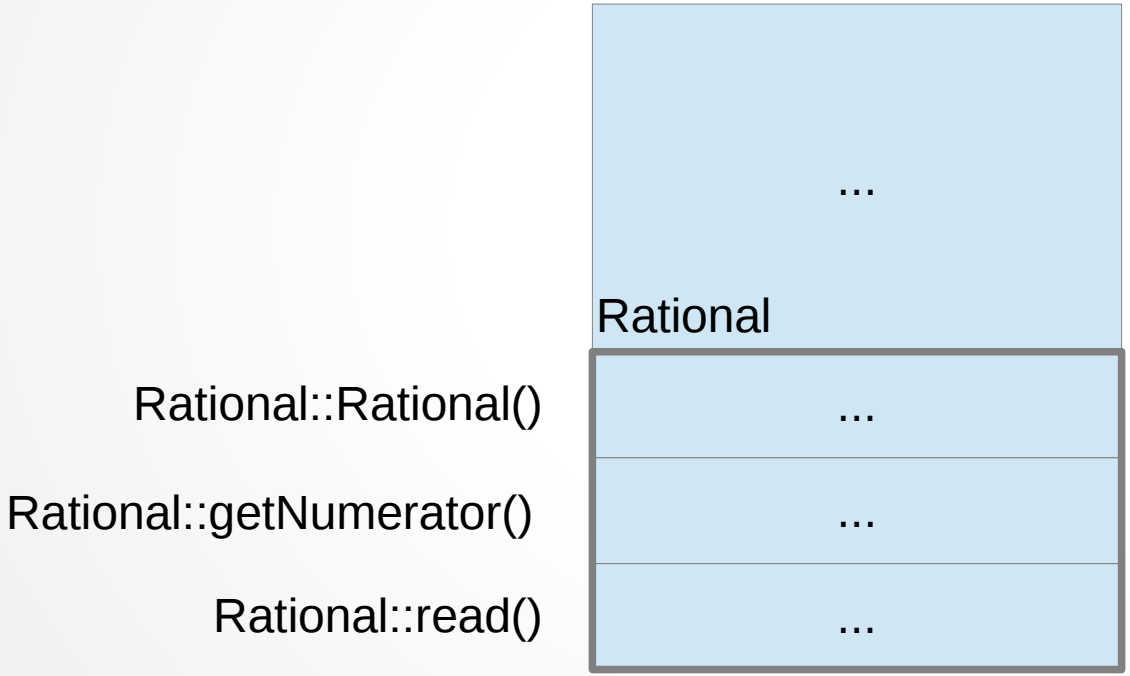
- Променливите от тип някой клас се наричат **обекти** или **инстанции на класа**
- <дефиниция_на_обект> ::= (<име_на_клас> | <клас>)
  <описание_на_обект> {, <описание_на_обект> };
- <описание на обект> ::=
  <име_на_обект> [= <израз>] |
  <име_на_обект>(<параметри>) |
  <име_на_обект> = <име_на_клас>(<параметри>)
-

Представяне на обекти в паметта



- всеки обект от даден клас заема едно и също количество памет
- sizeof(<клас>) или sizeof(<обект>) връщат броя байтове, заемани от обекта от дадения клас

Представяне на класове в паметта



Достъп до компонента на обект

- <обект>.<член-данна>
- <обект>.<член-функция>(<параметри>)
- Всеки обект има собствени стойности на член-данните
- Кодът на функциите се пази в класа, а не във всеки обект!

Достъп до компонента чрез указател към обект

- (*<указател_към_обект>).<член-данна>
- (*<указател_към_обект>).<член-функция>(<параметри>)
- <указател_към_обект>-><член-данна>
- <указател_към_обект>-><член-функция>(<параметри>)
- С указатели към обекти се работи както с указатели към обикновени променливи

Указател this

- В член-функциите имаме достъп до компонентите без да се указва обект
- Използва се обекта, за който е извикана член-функцията
- Как член-функциите разбират за кой обект са извикани?
- При всяко извикване на член-функция се създава автоматично константен указател
  <име_на_клас> * const this
- this винаги сочи към обекта, за който е извикана член-функцията
- За селекторите: <име_на_клас> const * const this

this като неявен параметър

- void Rational::read() {
  cin >> numer >> denom;
  }
  ... се превежда до
- void Rational::read(Rational* const this) {
  cin >> this->numer >> this->denom;
  }
- r.read();
  ... се превежда до
- Rational::read(&r);

this като неявен параметър

- int Rational::getNumerator() const {
  return numer;
  }
  ... се превежда до
- int Rational::getNumerator(Rational const * const this) {
  return this->numer;
  }
- cout << r.getNumerator();
  ... се превежда до
- cout << Rational::getNumerator(&r);

Спецификатори за достъп

Две нива на достъп:

- вътрешен достъп
  Достъп до компоненти на класа от член-функции на същия клас
- външен достъп
  Достъп до компоненти на класа от функции, които не са член-функции на същия клас
  - обикновени функции
  - член-функции на друг клас

Спецификатори за достъп

Спецификатори за достъп

- private
  - позволен е само вътрешен достъп
- public
  - позволен е вътрешен и външен достъп
- protected
  - позволен е вътрешен достъп и ограничен външен достъп
  - ще говорим за protected по-късно
- спецификатор по подразбиране е private
  - в struct е public

Спецификатори за достъп

- След първото използване на спецификатор за достъп, той остава валиден за всички последващи декларации
- Спецификатор за достъп може да бъде използван произволен брой пъти
- ```
  class Example {
    int a; // private
    double b; //private
  public:
    Example(); // public
    int getA() const; // public
  private:
    void setB(double b); // private
  };
  ```
  ```

Операция за указване на област

- [<област>]::<име>
- <област> е запис, клас или пространство от имена
- Ако <област> е пропусната, се подразбира глобалното пространство от имена
- Име с използвана операция :: се нарича **квалифицирано име**

Операция за указване на област

Примери:

- Rational::read — член-функцията read на класа Rational
- Student::read — член-функцията read на класа Student
- ::read — глобалната функция read
- Операцията :: се използва, когато има нужда да се разреши нееднозначност (ambiguity)

Дефиниция на член-функция

- <член_функция> ::=
  [inline] [<тип>] <име_на_клас>::<име_на_член_функция>
  (<параметри>) { <тяло> }
- Прието е член-функциите да се дефинират в изходния (source) файл, а не в заглавния (header) файл
- Защо?
- Който използва класа ви трябва да знае кои са член-данните и какви член-функции има, но не и как са реализирани

Вградени (inline) член-функции

- Позволено е член-функциите да се дефинират в дефиницията на класа class Rational {... Rational() { numer = 0; denom = 1; } };
- Такива функции се наричат **вградени**
- Вградените функции не се извикват със стекови рамки
- Тяхното тяло се замества при всяко тяхно извикване
- Една вградена функция може да е дефинирана извън дефиницията на класа
- Преди дефиницията се поставя запазената дума **inline**
- Окончателното решение дали една функция да е вградена е на компилатора!
- Препоръчително е да се вграждат само кратки функции

Примери за дефиниране на клас

- Точка в равнината
- Точка в пространството
- Пирамида