От записи към класове

(преговор с разширение)

Трифон Трифонов

Обектно-ориентирано програмиране,
спец. Компютърни науки, 1 поток,
спец. Софтуерно инженерство,
2016/17 г.

23 февруари 2017 г.

Абстракция със структури от данни

- Създаване на нови типове данни чрез групиране на съществуващи типове данни в запис (`struct`)

Абстракция със структури от данни

- Създаване на нови типове данни чрез групиране на съществуващи типове данни в запис (`struct`)
- Полетата на записа ще наричаме член-данни

Абстракция със структури от данни

- Създаване на нови типове данни чрез групиране на съществуващи типове данни в запис (`struct`)
- Полетата на записа ще наричаме член-данни
- Създаваме операции, които работят върху член-данните

Абстракция със структури от данни

- Създаване на нови типове данни чрез групиране на съществуващи типове данни в запис (`struct`)
- Полетата на записа ще наричаме член-данни
- Създаваме операции, които работят върху член-данните
  - операциите "знаят" как е представен обекта

Абстракция със структури от данни

- Създаване на нови типове данни чрез групиране на съществуващи типове данни в запис (`struct`)
- Полетата на записа ще наричаме член-данни
- Създаваме операции, които работят върху член-данните
  - операциите "знаят" как е представен обекта
- Външните функции, които използват новия тип, работят с него чрез операциите

Абстракция със структури от данни

- Създаване на нови типове данни чрез групиране на съществуващи типове данни в запис (`struct`)
- Полетата на записа ще наричаме член-данни
- Създаваме операции, които работят върху член-данните
  - операциите "знаят" как е представен обекта
- Външните функции, които използват новия тип, работят с него чрез операциите
  - външните функции "не знаят" как е представен обекта

Пример: тип "рационално число"

- **Логическо описание:** обикновена дроб

$$\mathbb{Q} = \{ \frac{p}{q}, p \in \mathbb{Z}, q \in \mathbb{N}^+ \}$$

числител, знаменател

$$\frac{2}{4} = \frac{1}{2}$$

Пример: тип "рационално число"

- **Логическо описание:** обикновена дроб
- **Физическо представяне:** запис с числител и знаменател

Пример: тип "рационално число"

- **Логическо описание:** обикновена дроб
- **Физическо представяне:** запис с числител и знаменател
- **Базови операции:**
  - конструиране на рационално число
  - получаване на числител
  - получаване на знаменател

Пример: тип “рационално число”

- **Логическо описание:** обикновена дроб
- **Физическо представяне:** запис с числител и знаменател
- **Базови операции:**
  - конструиране на рационално число
  - получаване на числител
  - получаване на знаменател
- **Аритметични операции:**
  - събиране, изваждане
  - умножение, деление
  - сравнение

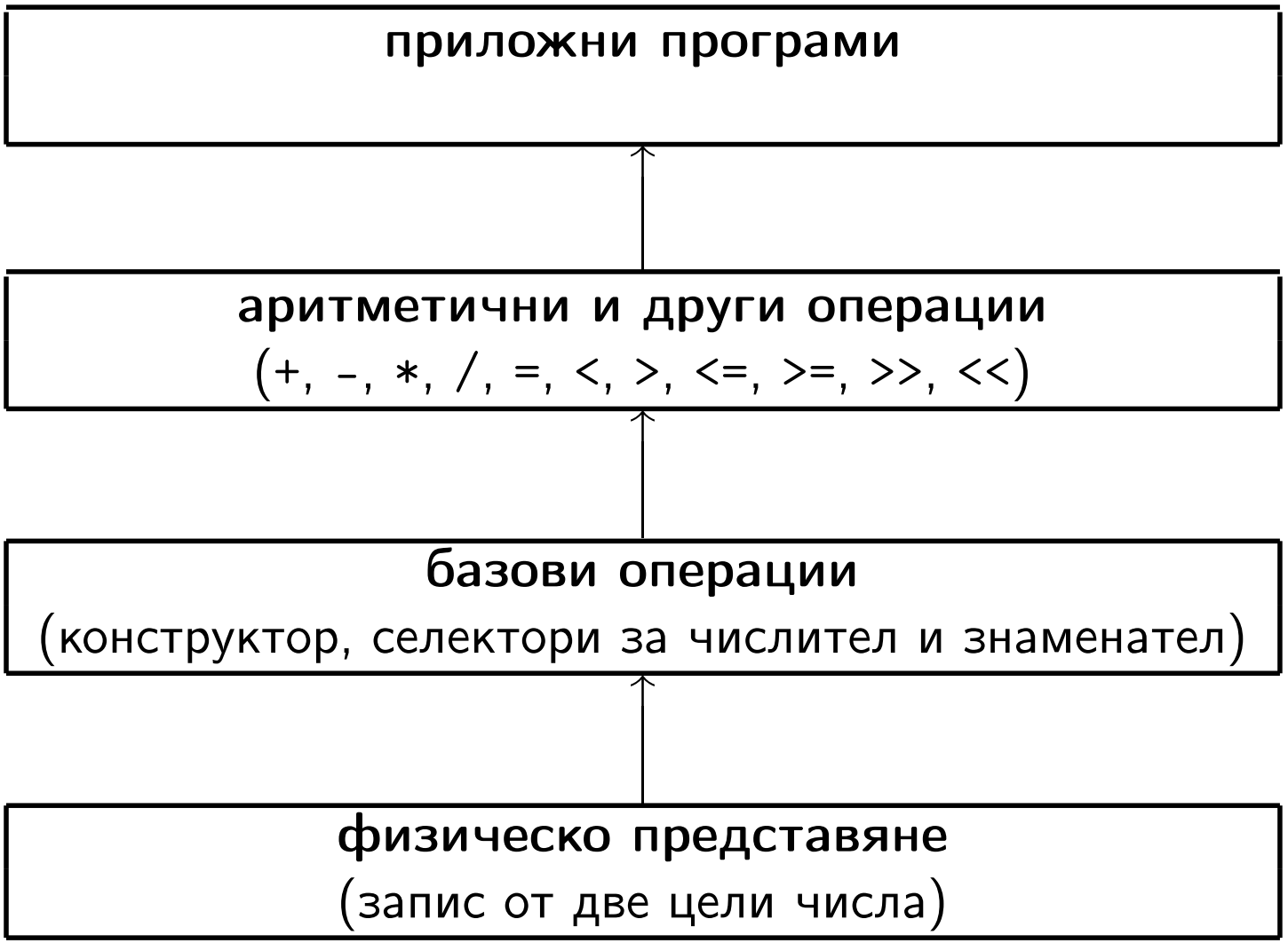
Пример: тип "рационално число"

- **Логическо описание:** обикновена дроб
- **Физическо представяне:** запис с числител и знаменател
- **Базови операции:**
  - конструиране на рационално число
  - получаване на числител
  - получаване на знаменател
- **Аритметични операции:**
  - събиране, изваждане
  - умножение, деление
  - сравнение
- **Други операции:**
  - въвеждане
  - извеждане
  - преобразуване до число с плаваща запетая

Пример: тип "рационално число"

- **Логическо описание:** обикновена дроб
- **Физическо представяне:** запис с числител и знаменател
- **Базови операции:**
  - конструиране на рационално число
  - получаване на числител
  - получаване на знаменател
- **Аритметични операции:**
  - събиране, изваждане
  - умножение, деление
  - сравнение
- **Други операции:**
  - въвеждане
  - извеждане
  - преобразуване до число с плаваща запетая
- **Приложни програми**

Нива на абстракция



Ниво 0: представяне на рационално число

```
struct Rational {
  int numer, denom;
};
```

Ниво 0: представяне на рационално число

```
struct Rational {
  int numer, denom;
};
```

- `numer` представя числителя на рационалното число
- `denom` представя знаменателя на рационалното число
- `numer` и `denom` са **член-данни** на новия тип данни `Rational`

Ниво 1: Конструктори

- Конструкторите са функции, които инициализират член-данните

Ниво 1: Конструктори

- Конструкторите са функции, които инициализират член-данните
- Конструкторите са "вътрешни функции" (**член-функции**, **методи**)

Ниво 1: Конструктори

- Конструкторите са функции, които инициализират член-данните
- Конструкторите са "вътрешни функции" (**член-функции**, **методи**)
- Конструкторите имат същото име като типа данни, който конструират

Ниво 1: Конструктори

- Конструкторите са функции, които инициализират член-данните
- Конструкторите са “вътрешни функции” (**член-функции**, **методи**)
- Конструкторите имат същото име като типа данни, който конструират
- Конструкторите винаги връщат типа данни, който конструират

Ниво 1: Конструктори

- Конструкторите са функции, които инициализират член-данните
- Конструкторите са "вътрешни функции" (**член-функции**, **методи**)
- Конструкторите имат същото име като типа данни, който конструират
- Конструкторите винаги връщат типа данни, който конструират
  - затова тип на резултата не се указва изрично

Ниво 1: Конструктори

- Конструкторите са функции, които инициализират член-данните
- Конструкторите са "вътрешни функции" (**член-функции**, **методи**)
- Конструкторите имат същото име като типа данни, който конструират
- Конструкторите винаги връщат типа данни, който конструират
  - затова тип на резултата не се указва изрично
- Конструктор по подразбиране (`Rational()`)

Ниво 1: Конструктори

- Конструкторите са функции, които инициализират член-данните
- Конструкторите са "вътрешни функции" (**член-функции**, **методи**)
- Конструкторите имат същото име като типа данни, който конструират
- Конструкторите винаги връщат типа данни, който конструират
  - затова тип на резултата не се указва изрично
- Конструктор по подразбиране (`Rational()`)
  - Използва се при дефиниране без изрична инициализация

Ниво 1: Конструктори

- Конструкторите са функции, които инициализират член-данните
- Конструкторите са "вътрешни функции" (**член-функции**, **методи**)
- Конструкторите имат същото име като типа данни, който конструират
- Конструкторите винаги връщат типа данни, който конструират
  - затова тип на резултата не се указва изрично
- Конструктор по подразбиране (`Rational()`)
  - Използва се при дефиниране без изрична инициализация
  - `Rational r;`

Ниво 1: Конструктори

- Конструкторите са функции, които инициализират член-данните
- Конструкторите са "вътрешни функции" (**член-функции**, **методи**)
- Конструкторите имат същото име като типа данни, който конструират
- Конструкторите винаги връщат типа данни, който конструират
  - затова тип на резултата не се указва изрично
- Конструктор по подразбиране (`Rational()`)
  - Използва се при дефиниране без изрична инициализация
  - `Rational r;`
  - `Rational r = Rational();`

Ниво 1: Конструктори

- Конструкторите са функции, които инициализират член-данните
- Конструкторите са "вътрешни функции" (**член-функции**, **методи**)
- Конструкторите имат същото име като типа данни, който конструират
- Конструкторите винаги връщат типа данни, който конструират
  - затова тип на резултата не се указва изрично
- Конструктор по подразбиране (`Rational()`)
  - Използва се при дефиниране без изрична инициализация
  - `Rational r;`
  - `Rational r = Rational();`

```
Rational() {
  numer = 0;
  denom = 1;
}
```

Ниво 1: Конструктори

- Конструктор с параметри (`Rational(n, d)`)

Ниво 1: Конструктори

- Конструктор с параметри (`Rational(n, d)`)
  - Приема по един параметър за всяка член-данна

Ниво 1: Конструктори

- Конструктор с параметри (`Rational(n, d)`)
  - Приема по един параметър за всяка член-данна
  - `Rational r(1, 2);`

Ниво 1: Конструктори

- Конструктор с параметри (`Rational(n, d)`)
  - Приема по един параметър за всяка член-данна
  - `Rational r(1, 2);`
  - `Rational r = Rational(1, 2);`

Ниво 1: Конструктори

- Конструктор с параметри (`Rational(n, d)`)
  - Приема по един параметър за всяка член-данна
  - `Rational r(1, 2);`
  - `Rational r = Rational(1, 2);`

```
Rational(int n, int d) {
  numer = n;
  denom = d;
}
```

Ниво 1: Селектори и мутатори

- Селекторите са **член-функции**, които позволяват преглед на член-данните

Ниво 1: Селектори и мутатори

- Селекторите са **член-функции**, които позволяват преглед на член-данните
- Селектори за достъп

Ниво 1: Селектори и мутатори

- Селекторите са **член-функции**, които позволяват преглед на член-данните
- Селектори за достъп
  - `int getNumerator()  { return numer; }`

Ниво 1: Селектори и мутатори

- Селекторите са **член-функции**, които позволяват преглед на член-данните
- Селектори за достъп
  - `int getNumerator()   { return numer; }`
  - `int getDenominator() { return denom; }`

Ниво 1: Селектори и мутатори

- Селекторите са **член-функции**, които позволяват преглед на член-данните
- Селектори за достъп
  - `int getNumerator()   { return numer; }`
  - `int getDenominator() { return denom; }`
- Селектори за извеждане/конвертиране

Ниво 1: Селектори и мутатори

- Селекторите са **член-функции**, които позволяват преглед на член-данните
- Селектори за достъп
  - `int getNumerator()   { return numer; }`
  - `int getDenominator() { return denom; }`
- Селектори за извеждане/конвертиране
  - `void print();`

Ниво 1: Селектори и мутатори

- Селекторите са **член-функции**, които позволяват преглед на член-данните
- Селектори за достъп
  - `int getNumerator()   { return numer; }`
  - `int getDenominator() { return denom; }`
- Селектори за извеждане/конвертиране
  - `void print();`
  - `double toDouble();`

Ниво 1: Селектори и мутатори

- Селекторите са **член-функции**, които позволяват преглед на член-данните
- Селектори за достъп
  - `int getNumerator()   { return numer; }`
  - `int getDenominator() { return denom; }`
- Селектори за извеждане/конвертиране
  - `void print();`
  - `double toDouble();`
- Мутаторите са **член-функции**, които позволяват промяна на член-данните

Ниво 1: Селектори и мутатори

- Селекторите са **член-функции**, които позволяват преглед на член-данните
- Селектори за достъп
  - `int getNumerator()   { return numer; }`
  - `int getDenominator() { return denom; }`
- Селектори за извеждане/конвертиране
  - `void print();`
  - `double toDouble();`
- Мутаторите са **член-функции**, които позволяват промяна на член-данните
- `void read();`

Ниво 2: Аритметични операции

Искаме да моделираме математическите операции над рационални числа:

Ниво 2: Аритметични операции

$$\frac{n_1}{d_1} + \frac{n_2}{d_2} = \frac{n_1 \cdot d_2 + n_2 d_1}{d_1 d_2}$$

Искаме да моделираме математическите операции над рационални числа:

- `Rational add(Rational p, Rational q);`
- `Rational subtract(Rational p, Rational q);`
- `Rational multiply(Rational p, Rational q);`
- `Rational divide(Rational p, Rational q);`

Ниво 3: Приложни програми

Задача. Да се намери рационално число, което приближава e^k.

Ниво 3: Приложни програми

Задача. Да се намери рационално число, което приближава e^k.
Решение. Ще напишем функция, която пресмята сумата:

$$\sum_{i=0}^{n} \frac{k^i}{i!}$$

Ниво 3: Приложни програми

Задача. Да се намери рационално число, което приближава e^k.
Решение. Ще напишем функция, която пресмята сумата:

$$\sum_{i=0}^{n} \frac{k^i}{i!}$$

Проблем. Числителите и знаменателите стават много големи!

Ниво 3: Приложни програми

Задача. Да се намери рационално число, което приближава e^k.
Решение. Ще напишем функция, която пресмята сумата:

$$\sum_{i=0}^{n} \frac{k^i}{i!}$$

Проблем. Числителите и знаменателите стават много големи!
Решение. Да използваме `long`.

Ниво 3: Приложни програми

Задача. Да се намери рационално число, което приближава e^k.
Решение. Ще напишем функция, която пресмята сумата:

$$\sum_{i=0}^{n} \frac{k^i}{i!}$$

Проблем. Числителите и знаменателите стават много големи!
Решение. Да използваме `long`.
Проблем. Само отлага препълването...

Ниво 3: Приложни програми

Задача. Да се намери рационално число, което приближава e^k.
Решение. Ще напишем функция, която пресмята сумата:

$$\sum_{i=0}^{n} \frac{k^i}{i!}$$

Проблем. Числителите и знаменателите стават много големи!
Решение. Да използваме `long`.
Проблем. Само отлага препълването...
Решение. Да съкращаваме дробите.

Ниво 3: Приложни програми

Задача. Да се намери рационално число, което приближава e^k.
Решение. Ще напишем функция, която пресмята сумата:

$$\sum_{i=0}^{n} \frac{k^i}{i!}$$

Проблем. Числителите и знаменателите стават много големи!
Решение. Да използваме `long`.
Проблем. Само отлага препълването...
Решение. Да съкращаваме дробите.
Проблем. На кое ниво да извършим съкращението?

Ниво 3: Приложни програми

Задача. Да се намери рационално число, което приближава e^k.
Решение. Ще напишем функция, която пресмята сумата:

$$\sum_{i=0}^{n} \frac{k^i}{i!}$$

Проблем. Числителите и знаменателите стават много големи!
Решение. Да използваме `long`.
Проблем. Само отлага препълването...
Решение. Да съкращаваме дробите.
Проблем. На кое ниво да извършим съкращението?
Решение. На възможно най-ниското: ниво 1.

Предимства на абстракцията

- Изолиране на промените

Предимства на абстракцията

- Изолиране на промените
  - промените по едно ниво налагат промени само в следващото ниво

Предимства на абстракцията

- Изолиране на промените
  - промените по едно ниво налагат промени само в следващото ниво
- Разпространяване на промените

Предимства на абстракцията

- Изолиране на промените
  - промените по едно ниво налагат промени само в следващото ниво
- Разпространяване на промените
  - подобренията в едно ниво се отразяват положително на всички по-горни нива

Предимства на абстракцията

- Изолиране на промените
  - промените по едно ниво налагат промени само в следващото ниво
- Разпространяване на промените
  - подобренията в едно ниво се отразяват положително на всички по-горни нива
- Ограничаване на знанието

Предимства на абстракцията

- Изолиране на промените
  - промените по едно ниво налагат промени само в следващото ниво
- Разпространяване на промените
  - подобренията в едно ниво се отразяват положително на всички по-горни нива
- Ограничаване на знанието
  - за реализацията на елемент на някое от нивата е нужна само информация за елементите на долното ниво

Предимства на абстракцията

- Изолиране на промените
  - промените по едно ниво налагат промени само в следващото ниво
- Разпространяване на промените
  - подобренията в едно ниво се отразяват положително на всички по-горни нива
- Ограничаване на знанието
  - за реализацията на елемент на някое от нивата е нужна само информация за елементите на долното ниво
  - не е нужно да познаваме в подробности как са реализирани операциите и алгоритмите за работа с член-данните

Предимства на абстракцията

- Изолиране на промените
  - промените по едно ниво налагат промени само в следващото ниво
- Разпространяване на промените
  - подобренията в едно ниво се отразяват положително на всички по-горни нива
- Ограничаване на знанието
  - за реализацията на елемент на някое от нивата е нужна само информация за елементите на долното ниво
  - не е нужно да познаваме в подробности как са реализирани операциите и алгоритмите за работа с член-данните
  - работата по отделните нива може да се извършва паралелно и независимо

Капсулация

Принцип на капсулацията:
Разделя се (абстрахира се) описанието на типа данни от конкретната му реализация.

Капсулация

Принцип на капсулацията:
Разделя се (абстрахира се) описанието на типа данни от конкретната му реализация.

- Описание (**интерфейс**): име на типа, сигнатури на функции и методи

Капсулация

Принцип на капсулацията:
Разделя се (абстрахира се) описанието на типа данни от конкретната му реализация.

- Описание (**интерфейс**): име на типа, сигнатури на функции и методи
- Представяне (**имплементация**): полета на типа, тела на функции и методи

Капсулация

Принцип на капсулацията:
Разделя се (абстрахира се) описанието на типа данни от конкретната му реализация.

- Описание (**интерфейс**): име на типа, сигнатури на функции и методи
- Представяне (**имплементация**): полета на типа, тела на функции и методи
- Абстракцията със структури от данни се възползва от капсулацията

Капсулация

Принцип на капсулацията:
Разделя се (абстрахира се) описанието на типа данни от конкретната му реализация.

- Описание (**интерфейс**): име на типа, сигнатури на функции и методи
- Представяне (**имплементация**): полета на типа, тела на функции и методи
- Абстракцията със структури от данни се възползва от капсулацията
  - представянето на ниво $n + 1$ работи с описанието на ниво n

Капсулация

Принцип на капсулацията:
Разделя се (абстрахира се) описанието на типа данни от конкретната му реализация.

- Описание (**интерфейс**): име на типа, сигнатури на функции и методи
- Представяне (**имплементация**): полета на типа, тела на функции и методи
- Абстракцията със структури от данни се възползва от капсулацията
  - представянето на ниво $n + 1$ работи с описанието на ниво n
  - представянето на ниво n не зависи от представянето на нивата $< n$

Капсулация

Принцип на капсулацията:
Разделя се (абстрахира се) описанието на типа данни от конкретната му реализация.

- Описание (**интерфейс**): име на типа, сигнатури на функции и методи
- Представяне (**имплементация**): полета на типа, тела на функции и методи
- Абстракцията със структури от данни се възползва от капсулацията
  - представянето на ниво $n + 1$ работи с описанието на ниво n
  - представянето на ниво n не зависи от представянето на нивата $< n$
- Предимства на капсулацията

Капсулация

Принцип на капсулацията:
Разделя се (абстрахира се) описанието на типа данни от конкретната му реализация.

- Описание (**интерфейс**): име на типа, сигнатури на функции и методи
- Представяне (**имплементация**): полета на типа, тела на функции и методи
- Абстракцията със структури от данни се възползва от капсулацията
  - представянето на ниво $n + 1$ работи с описанието на ниво n
  - представянето на ниво n не зависи от представянето на нивата $< n$
- Предимства на капсулацията
  - можем да подменим представянето без да се отрази на описанието

Капсулация

Принцип на капсулацията:
Разделя се (абстрахира се) описанието на типа данни от конкретната му реализация.

- Описание (**интерфейс**): име на типа, сигнатури на функции и методи
- Представяне (**имплементация**): полета на типа, тела на функции и методи
- Абстракцията със структури от данни се възползва от капсулацията
  - представянето на ниво $n + 1$ работи с описанието на ниво n
  - представянето на ниво n не зависи от представянето на нивата $< n$
- Предимства на капсулацията
  - можем да подменим представянето без да се отрази на описанието
  - външните функции зависят само от описанието

Капсулация

Принцип на капсулацията:
Разделя се (абстрахира се) описанието на типа данни от конкретната му реализация.

- Описание (**интерфейс**): име на типа, сигнатури на функции и методи
- Представяне (**имплементация**): полета на типа, тела на функции и методи
- Абстракцията със структури от данни се възползва от капсулацията
  - представянето на ниво $n + 1$ работи с описанието на ниво n
  - представянето на ниво n не зависи от представянето на нивата $< n$
- Предимства на капсулацията
  - можем да подменим представянето без да се отрази на описанието
  - външните функции зависят само от описанието
  - промени по представянето не налагат промени по външните функции

Капсулация

Принцип на капсулацията:
Разделя се (абстрахира се) описанието на типа данни от конкретната му реализация.

- Описание (**интерфейс**): име на типа, сигнатури на функции и методи
- Представяне (**имплементация**): полета на типа, тела на функции и методи
- Абстракцията със структури от данни се възползва от капсулацията
  - представянето на ниво $n + 1$ работи с описанието на ниво n
  - представянето на ниво n не зависи от представянето на нивата $< n$
- Предимства на капсулацията
  - можем да подменим представянето без да се отрази на описанието
  - външните функции зависят само от описанието
  - промени по представянето не налагат промени по външните функции
  - описанието обикновено е по-просто от представянето, изолира се сложността

Капсулация чрез заглавни (header) файлове

- В заглавните файлове се пише **интерфейс (описание)**:

Капсулация чрез заглавни (header) файлове

- В заглавните файлове се пише **интерфейс (описание)**:
  - дефиниции на записи и класове

Капсулация чрез заглавни (header) файлове

- В заглавните файлове се пише **интерфейс (описание)**:
  - дефиниции на записи и класове
  - декларации на функции

Капсулация чрез заглавни (header) файлове

- В заглавните файлове се пише **интерфейс (описание)**:
  - дефиниции на записи и класове
  - декларации на функции
  - декларации на методи

Капсулация чрез заглавни (header) файлове

- В заглавните файлове се пише **интерфейс (описание)**:
  - дефиниции на записи и класове
  - декларации на функции
  - декларации на методи
- В заглавните файлове обикновено не се пише **имплементация (представяне)**:

Капсулация чрез заглавни (header) файлове

- В заглавните файлове се пише **интерфейс (описание)**:
  - дефиниции на записи и класове
  - декларации на функции
  - декларации на методи
- В заглавните файлове обикновено не се пише **имплементация (представяне)**:
  - дефиниции на (глобални) променливи

Капсулация чрез заглавни (header) файлове

- В заглавните файлове се пише **интерфейс (описание)**:
  - дефиниции на записи и класове
  - декларации на функции
  - декларации на методи
- В заглавните файлове обикновено не се пише **имплементация (представяне)**:
  - дефиниции на (глобални) променливи
  - дефиниции на функции

Капсулация чрез заглавни (header) файлове

- В заглавните файлове се пише **интерфейс (описание)**:
  - дефиниции на записи и класове
  - декларации на функции
  - декларации на методи
- В заглавните файлове обикновено не се пише **имплементация (представяне)**:
  - дефиниции на (глобални) променливи
  - дефиниции на функции
  - дефиниции на методи

Капсулация чрез заглавни (header) файлове

- В заглавните файлове се пише **интерфейс (описание)**:
  - дефиниции на записи и класове
  - декларации на функции
  - декларации на методи
- В заглавните файлове обикновено не се пише **имплементация (представяне)**:
  - дефиниции на (глобални) променливи
  - дефиниции на функции
  - дефиниции на методи
- Имплементацията се пише в изходните (source) файлове

Капсулация чрез заглавни (header) файлове

- В заглавните файлове се пише **интерфейс (описание)**:
  - дефиниции на записи и класове
  - декларации на функции
  - декларации на методи
- В заглавните файлове обикновено не се пише **имплементация (представяне)**:
  - дефиниции на (глобални) променливи
  - дефиниции на функции
  - дефиниции на методи
- Имплементацията се пише в изходните (source) файлове
- Реферирането на методи извън типа става чрез оператора `::`

Капсулация чрез заглавни (header) файлове

- В заглавните файлове се пише **интерфейс (описание)**:
  - дефиниции на записи и класове
  - декларации на функции
  - декларации на методи
- В заглавните файлове обикновено не се пише **имплементация (представяне)**:
  - дефиниции на (глобални) променливи
  - дефиниции на функции
  - дефиниции на методи
- Имплементацията се пише в изходните (source) файлове
- Реферирането на методи извън типа става чрез оператора `::`
  - `Rational::Rational() { ... }`

Капсулация чрез заглавни (header) файлове

- В заглавните файлове се пише **интерфейс (описание)**:
  - дефиниции на записи и класове
  - декларации на функции
  - декларации на методи
- В заглавните файлове обикновено не се пише **имплементация (представяне)**:
  - дефиниции на (глобални) променливи
  - дефиниции на функции
  - дефиниции на методи
- Имплементацията се пише в изходните (source) файлове
- Реферирането на методи извън типа става чрез оператора `::`
  - `Rational::Rational() { ... }`
  - `void Rational::print() { ... }`

Капсулация чрез заглавни (header) файлове

- В заглавните файлове се пише **интерфейс (описание)**:
  - дефиниции на записи и класове
  - декларации на функции
  - декларации на методи
- В заглавните файлове обикновено не се пише **имплементация (представяне)**:
  - дефиниции на (глобални) променливи
  - дефиниции на функции
  - дефиниции на методи
- Имплементацията се пише в изходните (source) файлове
- Реферирането на методи извън типа става чрез оператора `::`
  - `Rational::Rational() { ... }`
  - `void Rational::print() { ... }`
- Внимание: Полетата на типа са част от представянето, но въпреки това се пишат в заглавния файл!

Капсулация чрез заглавни (header) файлове

- В заглавните файлове се пише **интерфейс (описание)**:
  - дефиниции на записи и класове
  - декларации на функции
  - декларации на методи
- В заглавните файлове обикновено не се пише **имплементация (представяне)**:
  - дефиниции на (глобални) променливи
  - дефиниции на функции
  - дефиниции на методи
- Имплементацията се пише в изходните (source) файлове
- Реферирането на методи извън типа става чрез оператора `::`
  - `Rational::Rational() { ... }`
  - `void Rational::print() { ... }`
- Внимание: Полетата на типа са част от представянето, но въпреки това се пишат в заглавния файл!
  - Компилаторът трябва да знае колко памет заемат променливите от дефинирания тип

Капсулация чрез спецификатори за достъп

- Спецификаторите за достъп са “етикети”, които указват как могат да се достъпват компонентите на типа (член-данни и член-функции)

Капсулация чрез спецификатори за достъп

- Спецификаторите за достъп са “етикети”, които указват как могат да се достъпват компонентите на типа (член-данни и член-функции)
- `private` — могат да бъдат достъпвани само от методите на типа

Капсулация чрез спецификатори за достъп

- Спецификаторите за достъп са “етикети”, които указват как могат да се достъпват компонентите на типа (член-данни и член-функции)
- `private` — могат да бъдат достъпвани само от методите на типа
- `public` — могат да бъдат достъпвани и от външни програми

Капсулация чрез спецификатори за достъп

- Спецификаторите за достъп са “етикети”, които указват как могат да се достъпват компонентите на типа (член-данни и член-функции)
- `private` — могат да бъдат достъпвани само от методите на типа
- `public` — могат да бъдат достъпвани и от външни програми
- препоръчително е всички член-данни да са капсулирани с `private`

Капсулация чрез спецификатори за достъп

- Спецификаторите за достъп са “етикети”, които указват как могат да се достъпват компонентите на типа (член-данни и член-функции)
- `private` — могат да бъдат достъпвани само от методите на типа
- `public` — могат да бъдат достъпвани и от външни програми
- препоръчително е всички член-данни да са капсулирани с `private`
- смислено ли е методи да са `private`?

Капсулация чрез спецификатори за достъп

- Спецификаторите за достъп са "етикети", които указват как могат да се достъпват компонентите на типа (член-данни и член-функции)
- `private` — могат да бъдат достъпвани само от методите на типа
- `public` — могат да бъдат достъпвани и от външни програми
- препоръчително е всички член-данни да са капсулирани с `private`
- смислено ли е методи да са `private`?
  - да, когато са за вътрешно ползване

Капсулация чрез спецификатори за достъп

- Спецификаторите за достъп са "етикети", които указват как могат да се достъпват компонентите на типа (член-данни и член-функции)
- `private` — могат да бъдат достъпвани само от методите на типа
- `public` — могат да бъдат достъпвани и от външни програми
- препоръчително е всички член-данни да са капсулирани с `private`
- смислено ли е методи да са `private`?
  - да, когато са за вътрешно ползване
  - пример: функция за съкращаване на рационални числа

От записи към класове

- Заменяме `struct` със `class`

От записи към класове

- Заменяме `struct` със `class`
- Какво се променя?

От записи към класове

- Заменяме `struct` със `class`
- Какво се променя?
  - Почти нищо!

От записи към класове

- Заменяме `struct` със `class`
- Какво се променя?
  - Почти нищо!
  - Класовете по подразбиране са капсулирани, а записите не са

От записи към класове

- Заменяме `struct` със `class`
- Какво се променя?
  - Почти нищо!
  - Класовете по подразбиране са капсулирани, а записите не са
  - В класовете нивото на достъп по подразбиране е `private`

От записи към класове

- Заменяме `struct` със `class`
- Какво се променя?
  - Почти нищо!
  - Класовете по подразбиране са капсулирани, а записите не са
  - В класовете нивото на достъп по подразбиране е `private`
  - В записите е `public`

От записи към класове

- Заменяме `struct` със `class`
- Какво се променя?
  - Почти нищо!
  - Класовете по подразбиране са капсулирани, а записите не са
  - В класовете нивото на достъп по подразбиране е `private`
  - В записите е `public`
- Написахме първия си клас!