# АБСТРАКЦИЯ

гл.ас. д-р. Нора Ангелова

# Абстракция

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

# Абстракция

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.

# Абстракция

Използването на подхода прави програмите по-лесни за описание и модификация.

Забелязваме, че ако се направи друг избор на представяне на данните, промяната ще се отрази единствено на реализациите на конструкторите, селекторите, мутаторите и предикатите.

# АБСТРАКЦИЯ

Проектиране за работа с абстрактни данни.

* *Проектира се да работи с „абстрактни данни“ - данни с неясно представяне.*

**Задача:**

Да се реализира умножение на рационални числа.

Резултатът от умножението на рационални числа?
n1/d1 * n2/d2 = n1*n2/d1*d2

*Нека имаме произволно представяне на рационално число - rat.*

1. Създаване на рационално число.
2. Извличане на числител на рационално число.
3. Извличане на знаменател на рационално число.

Да се реализира:

Функция за умножение на рационални числа.

# Абстракция

Създаване на рационално число

```
void makerat(rat& result, int n, int d);
```

Извличане на числител на рационално число

```
int numerator(rat& r);
```

Извличане на знаменател на рационално число

```
int denominator(rat& r);
```

Функция за умножение на рациални числа

```
rat multRats(rat& r1, rat& r2) {
  rat r;
  makerat(
    r,
    numerator(r1)*numerator(r2),
    denominator(r1)*denominator(r2)
  );
  return r;
}
```

# АБСТРАКЦИЯ

## Конкретизация на представянето

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

void makerat(rat& r, int n, int d) {
  r.num = n;
  r.denom = d;
}

int numerator(rat& r) {
  return r.num;
}

int denominator(rat& r) {
  return r.denom;
}
```

# Абстракция

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

rat абстрактен тип данни ли е?

** Множество от операции – могат да се реализират чрез член-функции на записа.*

# АБСТРАКЦИЯ

```
struct rat {
    int num;
    int denom;

    // член-функции на rat
    void makerat(int n, int d);
    int numerator();
    int denominator();
    void printRat();
};
```

# Достъп до член-функции

```
rat r;
r.makerat(1,5);
```

# АБСТРАКЦИЯ

```
void rat::makerat(int n, int d) {
    num = n;
    denom = d;
}

int rat::numerator() {
    return num;
}

int rat::denominator() {
    return denom;
}
```

# Достъп до член-данни

- Функцията за умножение на рационални числа не достъпва num && denom
- Ако се опитаме да ги достъпим.

```
rat r;
cout << r.num; // Опитът за достъп е успешен
```

- Може да се забрани чрез спецификатори за достъп.

# СПЕЦИФИКАТОРИ ЗА ДОСТЪП

- public – член-данните и член-функциите са достъпни за всяка функция, която е в обрастта на структурата.

  default

- private – член-данните и член-функциите са достъпни само за член-функциите.

- protected – ЩЕ СЕ РАЗГЛЕДА ПРИ НАСЛЕДЯВАНЕ.

# Спецификатори за достъп

```
struct rat {
  private:
  // член-данни на rat
  int num;
  int denom;

  public:
  // член-функции на rat
  void makerat(int n, int d);
  int numerator();
  int denominator();
  void printRat();
};
```

# Достъп

```
rat r;
r.makerat(1,5); // Има достъп до член-функцията
cout << r.num;  // Няма достъп до член-данната. Грешка!
```

# КРАЙ

?