14. Rekurzija

1

REKURZIVNE FUNKCIJE

Kaj je rekurzija?

2

- funkcija je **rekurzivna**, če kliče sama sebe
- reukrzivno reševanje problema P
  - rešiti moramo problem P za dano množico podatkov N
    - REKURZIVNA ZVEZA: uporabimo rešitev problema P za manjšo množico podatkov N'
    - USTAVITVENI POGOJ: za zelo majhen N problem P rešimo direktno

Primer 01 – Fibonaccijevo zaporedje 1/2

- Fibonaccijevo zaporedje
  - $a_0=0$
  - $a_1=1$
  - $a_n= a_{n-1} + a_{n-2}$, za $n>1$
  - prvih nekaj členov: 0, 1, 1, 2, 3, 5, 8, 13, 21, ...
- razberimo rekurzivno zvezo in ustavitveni pogoj:
  - rekurzivna zveza: $a_n= a_{n-1} + a_{n-2}$
  - ustavitveni pogoj: $n=0$ ali $n=1$ poznamo rezultat

Primer 01 – Fibonaccijevo zaporedje 2/2


```
#include <iostream>
using namespace std;

// izračuna n-ti člen Fibonaccijevega zaporedja
int fib(int n) {
  if (n==0) return 0;
    else if (n==1) return 1;
      else return fib(n-1) + fib(n-2);
}

int main()
{
    int i;
    cout << "Vnesi nenegativno celo stevilo: ";
    cin >> i;
    cout << i <<". clen Fibonaccijevega zaporedja je " << fib(i);
    return 0;
}
```

Primer 02 – potenca števila 1/2

5

- napiši rekurzivno funkcijo, ki izračuna potenco nekega števila z naravnim eksponentom

- $a^n = a \cdot a \cdot \ldots \cdot a$ (n-krat množimo a)
  - ni rekurzivna zveza, saj nismo pri izračunu nikjer znova uporabili potenciranja
- $a^n = a^{n-1} \cdot a$
  - rekurzivna zveza, pri izračunu uporabimo potenciranje, vendar za manjši eksponent (manjšo množico podatkov)
  - ustavitveni pogoj:
    - n=1 saj takrat enostavno izračunamo rezultat: $a^1 = a$

Primer 02 – potenca števila 2/2

6

```
#include <iostream>
using namespace std;

double pot(double a, int n) {
    if (n==1)
        return a;
    else
        return pot(a, n-1) * a;
}

int main()
{
    double x; int y;
    cout << "Vnesi osnovo (poljubno stevilo): ";
    cin >> x;
    cout << "Vnesi eksponent (naravno stevilo): ";
    cin >> y;
    cout << x << "^" << y << " je " << pot(x,y);
    return 0;
}
```

Še o rekurziji

7

- rekurzivne rešitve so običajno počasne
- vsako rekurzivno rešitev lahko zapišemo v nerekurzivni obliki
- z rekurzijo je povezan pojem **sklad**
  - sklad – podatkovna struktura
  - deluje po principu LIFO (angl. Last In First Out)
  - na skladu se hranijo podatki (spremenljivke, parametri, mesto klica) o funkciji, ki kliče drugo funkcijo (tudi samo sebe)
  - kadar želimo rekurzivno funkcijo zapisati v nerekurzivni obliki in nimamo druge ideje, lahko simuliramo situacijo s skladom

Vaje iz rekurzije

8

Spodnje naloge reši na rekurziven način:

1. Napiši funkcijo, ki izračuna vsoto števk nekega nenegativnega celega števila.
2. Napiši funkcijo, ki izračuna število števk nekega nenegativnega celega števila.
3. Napiši funkcijo, ki izpiše niz od zadaj naprej.