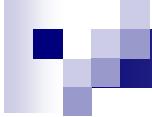


Programiranje I – RIN Računalništvo I – MA

REKURZIVNI ZAPIS FUNKCIJ



Algoritem

■ Kaj je algoritem?

1. To je funkcija, ki vhodne podatke pretvori v izhodne - nekaj izračuna
2. Algoritem izračuna rezultat za **vsake** podatke - funkcija se vedno ustavi

Primer

```
public int sestejDo0(int par) {  
    int vsota;  
    vsota = 0;  
    while (par != 0) {  
        vsota += par;  
        par--;  
    }  
    return vsota;  
} /* sestejDo0 */
```

- Ali je to algoritem?



Seštevanje števil do 0

```
Int sestejDo0(int par) {  
    int vsota;  
    vsota = 0;                      /* inicializacija na 0 */  
    while (par != 0) {                /* dokler je par več od 0 */  
        vsota += par;                /* vsoti prištevamo par, */  
        par--;                      /* ki ga ob vsakem koraku  
                                     zmanjšamo za 1 */  
    }  
    return vsota;  
} /* sestejDo0 */
```

Seštevanje malce drugače

```
Int sestejDo0(int par) {      Int sestejDo0(int par) {
    int vsota;
    vsota = 0;
    if (par <= 0)                  if (par <= 0)
        return 0;                  return 0;
    else {
        while (par != 0) {
            vsota += par;
            par--;
        }
        return vsota;              return
    }                                sestejDo0(par-1)+par;
} /* sestejDo0 */                  } /* sestejDo0 */
```

Kako v resnici seštevamo

$$\text{rezultat} = (\sum_{i=0..par} i) = (\sum_{i=0..par-1} i) + par$$

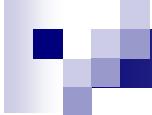
```
Int sestejDo0(int par) {    Int sestejDo0(int par) {  
    če je par manjši ali        if (par <= 0)  
    enak nič, potem            return 0;  
    je rezultat 0;  
    sicer  
    je rezultat enak  
    seštevku od 0 do par-1,  
    ki mu prištejemo par        return  
    } /* sestejDo0 */                sestejDo0(par-1)  
                                    + par;  
} /* sestejDo0 */
```

Rekurzivni zapis

```
Int sestejDo0(int par) {  
    if (par <= 0)  
        return 0;  
    else  
        return  
            sestejDo0(par-1)+par;  
} /* sestejDo0 */
```

```
Int sestejDo0(int par) {  
    if (par <= 0)  
        je rezultat 0;  
    else  
        je rezultat vsota  
        rezultata manjšega  
        problema in par  
} /* sestejDo0 */
```

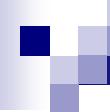
- Ustavitevni pogoj
- Korak deli in vladaj



Ponovno - algoritem

```
Int sestejDo0(int par) {  
    if (par <= 0)  
        return 0;  
    else  
        return  
            sestejDo0(par-1)+par;  
} /* sestejDo0 */
```

- Ali je rekurzivno zapisana funkcija algoritem?
- Da, ker se ustavi za vsako vrednost par .
- *Dokaz* (z indukcijo po parametru par):
 - *Osnova*: za vsak $\text{par} \leq 0$ je zaradi **kode**
 - *Hipoteza*: če je trditev resnična za $\text{par} = n-1$, potem je resnična tudi za $\text{par} = n$
 - *Korak*: recimo, da se funkcija ustavi za $\text{par}-1$; potem se vsekakor ustavi tudi za par (**zaradi kode**)



Rekurzija in iteracija

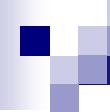
- Vsako ponavljanje (iteracijo) lahko zapišemo kot rekurzijo

```
public static void print(int[] polje) {  
    for(int i = 0; i < polje.length; i++)  
    {  
        System.out.print(polje[i]);  
        System.out.print(" ");  
    }  
    System.out.println();  
} // print
```

Izpis elementov polja

```
public static void print(int[] polje) {  
    for(int i = 0; i < polje.length; i++)  
        System.out.print(polje[i] + " ");  
    System.out.println();  
} // print
```

```
public static void print(int[] polje, int i) {  
    if (i >= polje.length)  
        System.out.println();  
    else {  
        System.out.print(polje[i] + " ");  
        print(polje, i+1);  
    }  
} // print
```



Izpis elementov polja

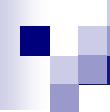
- če smo prišli do konca polja, izpišemo zaključek vrstice; sicer
- izpišemo naslednji znak (prvi v trenutnem polju) in preostanek polja

```
public static void print(int[] polje, int i)
{
    if (i >= polje.length)
        System.out.println();
    else {
        System.out.print(polje[i] + " ");
        print(polje, i+1);
    }
} // print
```

Iskanje elementa

```
public static boolean
member(int[] polje, int elt) {
    for(int i = 0; i < polje.length; i++)
        if (polje[i] == elt) return true;
    return false;
} // member

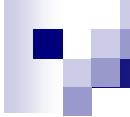
public static boolean
member(int[] polje, int elt, int i) {
    if (polje[i] == elt)      return true;
    if (i >= polje.length)   return false;
    return member(polje, elt, i+1);
} // member
```



Iskanje elementa

- če smo znak našli, vrnemo *true*
- če smo prišli do konca polja, vrnemo *false*
- sicer pregledamo preostanek polja

```
public static boolean  
member(int[] polje, int elt, int i) {  
    if (polje[i] == elt)      return true;  
    if (i >= polje.length)  return false;  
    return member(polje, elt, i+1);  
} // member
```



Iskanje največjega elementa

- če je zmanjkalo polja, vrnemo doslej največji najdeni element; sicer
- pogledamo ali je trenutni element večji od doslej najdenega elementa in, če je, ga proglasimo za trenutno največji element
- potem pogledamo še v preostanku polja, ali je kakšen večji element

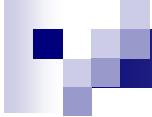
Iskanje največjega elementa

```
public static  
int maximum(int[] polje, int i, int trenMaks) {  
    if (i >= polje.length) return trenMaks;  
    if (polje[i] > trenMaks) trenMaks = polje[i];  
    return maximum(polje, i+1, trenMaks);  
} // maximum
```

- Pokličemo takole:

maximum (polje, 0, polje[0])

Kaj pa takole **maximum (polje, 1, polje[0])**?



Iskanje največjega elementa - 2.

- če je zadnji element v polju, potem je to tudi največji element v polju in ga vrnemo; sicer
- poiščemo največji element v preostanku polja
- pogledamo ali je trenutni element večji od doslej največjega elementa v preostanku polja in, če je, ga proglašimo za trenutno največji element



Iskanje največjega elementa - 2.

```
public static int maximum(int[] polje, int i) {  
    int maksPreost;  
    if ((i+1) == polje.length) return polje[i];  
    maksPreost = maximum(polje, i+1);  
    if (polje[i] > maksPreost) maksPreost = polje[i];  
    return maksPreost;  
} // maximum
```

- Pokličemo takole:

maximum (polje, 0)