Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

# Uvod v računalništvo

22.11 – 28. 11. 2021

Vaje 7

# Programski jeziki

- Proceduralni
  - FORTRAN, COBOL, C, C++, Java, C#, Python...
- Namenski
  - SQL, HTML, JavaScript...
- Alternativne paradigme
  - Funkcijsko programiranje: LISP, FP, Scheme...
  - Logično programiranje: Prolog...
  - Paralelno programiranje.

# Primer: proceduralno (Python) in funkcijsko (Scheme) programiranje

a) $f(x) = 2x$

b) $f(x) = x^2$

c) $f(x) = 2x^2$

d) $f([x_1, x_2, \ldots x_n]) = x_1 + x_2 + \ldots + x_n$

# Primer: proceduralno (Python) in funkcijsko (Scheme) programiranje

```
def double(x):
    return 2*x
def square(x):
    return x*x:
def poly(x):
    return double(square(x))
def adder(in_list):
    sum = 0
    for x in in_list:
        sum += x
    return sum
```

```
(define (double x) (*2 x))
(define (square x) (*x x))
(define (poly x) (double(square x )))
car list – vrne prvi element niza
cdr list – vrne niz prez prvega elementa
null? list – true če je niz prazen
cond ( (pogoj) ….) (else …)
(define adder (in_list)
   (cond ((null? In_list) 0)
      (else (+  (car in_list)  (adder (cdr in_list) ) ) ) ) )
```

# Primer: logično programiranje (Prolog)

- Dejstva

male(eli)
male(bill)
male(joe)
female(mary)
female(betty)
female(sarah)
parent-of(eli, bill)
parent-of(mary, bill)
parent-of(bill, joe)
parent-of(bill, betty)
parent-of(bill, sarah)

- Pravila

- Poizvedba

# Primer: logično programiranje (Prolog)

- Dejstva

male(eli)
male(bill)
male(joe)
female(mary)
female(betty)
female(sarah)
parent-of(eli, bill)
parent-of(mary, bill)
parent-of(bill, joe)
parent-of(bill, betty)
parent-of(bill, sarah)

- Pravila

father-of(X,Y):-parent-of(X,Y),male(X)

daughter-of(X,Y):-parent-of(Y,X),female(X)

- Poizvedba

?-parent-of(X,bill)

X = eli

X = mary

?-daughter-of(X,bill)

X = betty

X = sarah

# Algoritmi

- Definicija
  - Popolnoma urejeno zaporedje nedvoumnih in učinkovito izračunljivih operacij, ki, ko se le-te izvedejo, proizvede rezultat in se ustavi v končnem času.
  - Računalništvo je veda o načrtovanju in razvoju algoritmov za reševanje množice pomembnih problemov.
- Primeri
  - navodila za sestavljanje igrače
  - recept za peko torte
  - navodila za sestavljanje omare
  - postopek za izračun fakultete števila
  - postopek za izračun histograma slike
  - postopek za seštevanje dveh večmestnih števil
  - postopek za pretvarjanje iz decimalnega v dvojiški zapis
- Tri kategorije operacij
  - Zaporedne operacije
  - Pogojne operacije
  - Iterativne operacije

# Naloga 8.1

Navodila za dodajanje oseb v imenik na telefonu bi lahko izgledala nekako takole.

Korak 1. Če želite ustvariti skupino, sledite postopku na str. 7 preden nadaljujete s korakom 2.
Korak 2. V meniju izberite možnost "Imenik".
Korak 3. Ponovite korake od 4. do 7. za vsako osebo, ki jo želite dodati v imenik.
Korak 4. V meniju izberite možnost "Vnesi osebo".
Korak 5. Vnesite ime osebe.
Korak 6. Pomaknite se eno polje navzdol.
Korak 7. Vnesite telefonsko številko osebe.
Korak 8. Izberite možnost "Vnos končan".
Korak 9. Dodajanje oseb je zaključeno. Za spreminjanje podatkov o že vnesenih osebah sledite navodilom na strani 15.

Kakšne vrste operacij opaziš v korakih tega algoritma? Ali je operacija v koraku 6 nedvoumna?
Kaj pa operacije v vseh ostalih korakih? Utemelji.

# Naloga 8.2

Spodaj je zapisan algoritem za seštevanje dveh m-mestnih števil.

*Podano: dve pozitivni celi števili, vsako izmed njiju vsebuje m števk,* $a_{m-1}a_{m-2}...a_0$ *in* $b_{m-1}b_{m-2}...b_0$. *Pri tem velja* m ≥1.

*Dobiti želimo:* $c_mc_{m-1}c_{m-2}...c_0$, *kjer* $c_mc_{m-1}c_{m-2}...c_0 = (a_{m-1}a_{m-2}...a_0) + (b_{m-1}b_{m-2}...b_0)$.

Algoritem:

Korak 1. Nastavi vrednost spremenljivke *prenos* na 0.

Korak 2. Nastavi vrednost spremenljivke *i* na 0.

Korak 3. Dokler je vrednost spremenljivke *i* manjša ali enaka *m-1*, ponovi navodila v korakih od 4 do 6.

Korak 4. Izračunaj $c_i$ tako, da sešteješ števki $a_i$ in $b_i$ ter trenutno vrednost spremenljivke *prenos*.

Korak 5. Če ci ≥ 10,nastavi $c_i$ na $(c_i - 10)$ in nastavi vrednost spremenljivke *prenos* na 1; sicer nastavi vrednost spremenljivke *prenos* na 0.

Korak 6. Prištej 1 spremenljivki *i* (pomik en stolpec v levo).

Korak 7. Nastavi $c_m$ na vrednost spremenljivke *prenos*.

Korak 8. Izpiši končni rezultat: $c_mc_{m-1}c_{m-2}...c_0$.

Korak 9. Ustavi se.

Korak za korakom izvedi algoritem za seštevanje na številih 119 in 22. Na vsakem koraku zapiši vrednosti *c3*, *c2*, *c1*, *c0* in *prenos*.

# Naloga 8.2

# Naloga 8.3

Algoritem iz naloge 8.2 spremeni tako, da ne bo izpisal začetnih ničel. Odgovor na vprašanje iz naloge 4 naj bi torej bil 141 namesto 0141.

# Naloga 8.4

Eden izmed možnih načinov izvedbe množenja števil je tudi večkratno seštevanje. Na primer, zmnožek 17 x 20 lahko izračunamo tudi kot 17 + 17 + 17 … + 17 (dvajset krat). Podajte algoritem za množenje dveh pozitivnih celih števil, ki bo temeljil na zgornji zamisli.

# Naloga 8.5

Podan je algoritem za iskanje največjega števila v seznamu:

Preberi vrednost *n* (velikost seznama).
Preberi vrednosti $A_1$, $A_2$, …, $A_n$ (seznam števil).
Nastavi vrednost *doSedajNajvecji* na $A_1$.
Nastavi vrednost *polozaj* na 1.
Nastavi vrednost *i* na 2.
Dokler (i ≤ n), ponavljaj
  Če $A_i$ > *doSedajNajvecji*
    Nastavi *doSedajNajvecji* na vrednost $A_i$.
    Nastavi *polozaj* na vrednost *i*.
  *i* povečaj za 1.
Konec zanke.
Izpiši vrednosti *doSedajNajvecji* in *polozaj*.
Ustavi se.

Zgornji algoritem je bil razvit ob predpostavki, da so vsa števila v seznamu različna. Kaj se zgodi v primeru, če vsa števila niso različna?

Ali bi algoritem našel prvo pojavitev največjega števila, zadnjo pojavitev ali morda vsako pojavitev? Razloži obnašanje algoritma v danem primeru.

# Naloga 8.6

V šesti vrstici algoritma za iskanje največjega števila (naloga 8.5) imamo operacijo

*Dokler (i ≤ n), ponavljaj*

Razloži, kaj bi se zgodilo, če bi zgornjo operacijo spremenili v

- *Dokler (i ≥ n), ponavljaj*
- *Dokler (i < n), ponavljaj*
- *Dokler (i = n), ponavljaj*

# Naloga 8.7

Napiši algoritem, ki kot vhod prejme pozitivno celo število N in ugotovi, ali je N praštevilo (t.j. naravno število večje od 1, ki ima le dva pozitivna delitelja, število 1 in samega sebe). Kot izhod naj algoritem izpiše ustrezno sporočilo: "podano število JE praštevilo" oziroma "podano število NI praštevilo".

# Naloga 8.8

Napiši algoritem, ki kot vhod dobi seznam k celih števil $N_1$, $N_2$, … $N_k$, prav tako pa tudi celo število VST. Algoritem mora najti dvojico števil na seznamu, ki skupaj tvorita vsoto VST. Recimo, da imamo seznam s števili 3, 8, 13, 2, 17, 18 in 10, vrednost VST pa je 20. V tem primeru naj bi algoritem na izhodu vrnil dvojico (2, 18) ali pa dvojico (3, 17). Če v seznamu ni ustrezne dvojice, naj algoritem izpiše sporočilo "V danem seznamu ni tovrstne dvojice števil".

# Domača naloga - 1

V sedmi vrstici algoritma za iskanje največjega števila (naloga 8.5) imamo operacijo

$$\text{Če } A_i > doSedajNajvecji$$

Razloži, kaj bi se zgodilo, če bi zgornjo operacijo spremenili v

- Če $A_i \geq doSedajNajvecji$
- Če $A_i < doSedajNajvecji$

Kaj lahko poveš o pomembnosti uporabe pravilnih primerjalnih operacij (<, >, ≤, ≥, =, ≠) pri pisanju pogojnih in iterativnih operacij?

# Domača naloga - 2

Spodaj je predstavljen Evklidov algoritem za iskanje največjega skupnega delitelja dveh pozitivnih celih števil *I* in *J* :

Korak 1. Kot vhod dobiš dve pozitivni celi števili. Večje izmed njiju označi z *I*, manjše pa z *J*.

Korak 2. Deli *I* z *J*, ostanek označi z *R*.

Korak 3. Če *R* ni enak 0, nastavi *I* na vrednost enako *J*, *J* nastavi na vrednost enako *R* in pojdi na korak 2.

Korak 4. Izpiši odgovor, ki je kar enak vrednosti *J*-ja.

Korak 5. Ustavi se.

a) Pojdi skozi algoritem z vhodnima številoma 20 in 32. Po koncu vsakega koraka zabeleži vrednosti *I*, *J* in *R*. Poišči končni odgovor algoritma.

b) Ali algoritem deluje pravilno pri vhodu 0 in 32? Opiši, kaj točno se zgodi in spremeni algoritem tako, da vrne primerno sporočilo o napaki.

# Domača naloga - 3

Napiši algoritem za izračun tedenskega honorarnega plačila za delo. Kot vhod algoritem dobi število opravljenih delovnih ur v tednu in urno postavko. Končno plačilo naj se določi tako, da se za vse ure do dopolnjene 40. ure upošteva osnovno (podano) urno postavko, za ure od dopolnjene 40. do dopolnjene 54. količnik 1,50 urne postavke in za ure od dopolnjene 54. dalje količnik 2,00 urne postavke. Algoritem naj izračuna in izpiše višino plačila. Po izpisu plačila naj uporabnika tudi vpraša, ali želi opraviti naslednji izračun. Celoten postopek naj se ponavlja, dokler uporabnik ne odgovori z "NE".

# Domača naloga - 4

Podana je funkcija:

```
(define (mystery input-list)
   (cond ((null? input-list) 0)
         (else ( + 1 (mystery (cdr  input-list))))))
```

a) Kaj bo rezultat ukaza **(mystery (list 3 4 5))** ?

b) Razložite kaj na splošnem dela funkcija.

# Hvala za pozornost!

matej.dobrevski@fri.uni-lj.si