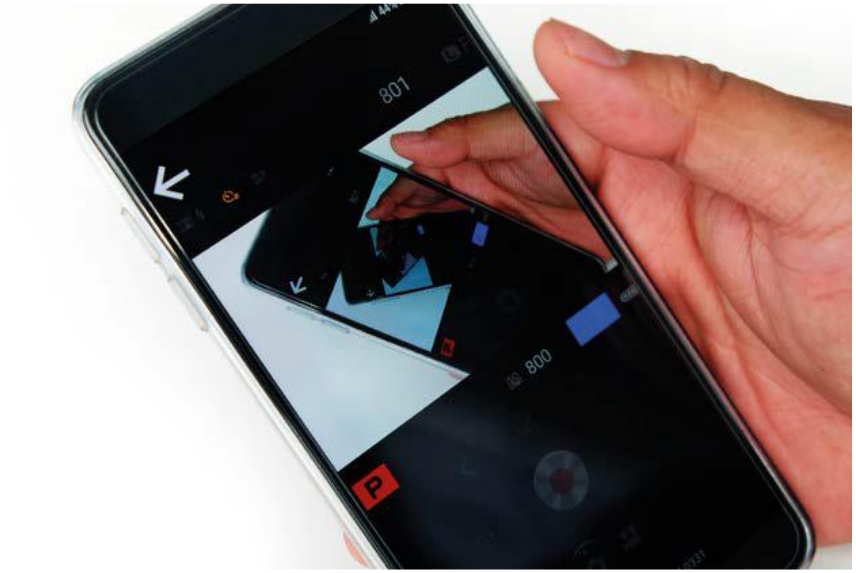
3. RAČUNALNO RAZMIŠLJANJE

3.1 Rekurzije

SAMOSLIČNOST

- **Samosličnost** je svojstvo objekta da sadrži dijelove u manjem mjerilu koji su slični cjelini.



ZADATAK

Riješi problem.

- Nora je voditeljica plesne skupine i nakon održane audicije mora izabrati *n* plesača/ plesačica koji će sudjelovati na natjecanju.
- Napiši algoritam za rješavanje tog problema.

RJEŠAVANJE PROBLEMA

1. Razumijevanje problema
2. Stvaranje plana rješavanja problema
3. Izvršavanje osmišljenog plana
4. Osvrt na rješenje i metodu rješavanja

1. RAZUMIJEVANJE PROBLEMA

- Među plesačima jedne skupine Nora mora izabrati *n* plesača/plesačica.
- Naravno, broj *n* manji je od ukupnog broja plesača/plesačica koji/koje su sudjelovali/sudjelovale na audiciji.

2. STVARANJE PLANA RJEŠAVANJA PROBLEMA

- Želi li Nora imati skupinu plesača, mora birati jednog po jednog plesača.
- Ponavljanje u algoritmu može se ostvariti dvojako. Prvi je način uporabom **petlje (iteracijom)** u kojoj će se *n* puta ponoviti naredba *Odaberi plesača/plesačicu.*
- **Iteracija (petlja)** je algoritamski postupak ponavljanja naredbi određeni, konačni broj puta.

Algoritam:

Za i = 1 do n ponovi

    Odaberi plesača/plesačicu

sljedeći i

3. IZVRŠAVANJE OSMIŠLJENOG PLANA

Algoritam:

Ples (n)

Ako je n=0 onda stani

inače

Odaberi plesača/plesačicu

Ples (n-1)

1. Algoritam *Ples* ima ulazni podatak *n* koji označava broj plesača koje treba izabrati.

2. Zaustavljanje izvršavanja algoritma

3. Funkcija *Ples* poziva samu sebe s izmijenjenom (manjom) ulaznom vrijednošću.

4. OSVRT NA RJEŠENJE I METODU RJEŠAVANJA

- Algoritam kojim je riješen problem naziva se **rekurzivni algoritam** ili **rekurzija**.
- Osnovna je ideja rekurzije riješiti algoritam funkcijom koja poziva samu sebe. U ovom slučaju to je funkcija *Ples*.
- Unutar algoritma mora postojati naredba kojom se prekida izvršavanje algoritma.
  U protivnom, budući da funkcija poziva samu sebe, algoritam bi se izvodio beskonačno.
  Ovaj algoritam završava kada ulazna vrijednost funkcije *Ples* postane *0*, odnosno kad su izabrani svi plesači.

REKURZIVNI ALGORITAM

- **Rekurzivni algoritam** ili rekurzija je algoritamski postupak u kojem rekurzivna funkcija poziva samu sebe.
- Algoritam završava kada ulazna vrijednost funkcije (u ovom slučaju Ples) postane 0. Taj dio algoritma naziva se **osnovni slučaj**.
- Dio algoritma u kojem funkcija poziva samu sebe naziva se **rekurzivni poziv**.
- **Stog** je struktura podataka u koju se pohranjuju varijable rekurzivne funkcije koja funkcionira na LIFO način rada.
- **LIFO** (Last In First Out) način rada znači da podataka koji je zadnji ušao prvi izlazi van.

ZADATAK

Riješi problem.

- Za učitani broj *n* napiši rekurzivni algoritam za izračunavanje umnoška prirodnih brojeva do *n*.

1. RAZUMIJEVANJE PROBLEMA

Ako je n = 1, umnožak će biti 1

Ako je n = 2, umnožak će biti 1 · 2 = 2

Ako je n = 3, umnožak će biti 1 · 2 · 3 = 6

Ako je n = 4, umnožak će biti 1 · 2 · 3 · 4 = 24

Za bilo koji prirodni broj *n* umnožak će biti jednak 1 · 2 · 3 · 4 · … · (n - 1) · n

Možeš uočiti da svaki sljedeći umnožak dobiješ kao umnožak broja *n* i umnoška prethodnih *n - 1* brojeva.

2. STVARANJE PLANA RJEŠAVANJA PROBLEMA

- Možeš uočiti da se u ovom problemu izračunavanja umnoška zadanih brojeva ponavlja postupak množenja zadanog broja *n* s umnoškom prethodnih *n - 1* prirodnih brojeva sve dok taj prirodni broj *n* ne bude jednak 1.
- Na primjer, za zadani broj n = 5 postupak će biti sljedeći:

umnožak (5) = 5 · umnožak (4)

= 5 · 4 · umnožak (3)

= 5 · 4 · 3 · umnožak (2)

= 5 · 4 · 3 · 2 · umnožak (1)

= 5 · 4 · 3 · 2 · 1 = 120

Znači, ako je n = 1 umnožak je 1,

a inače je umnožak (n) = n · umnožak (n - 1).

3. IZVRŠAVANJE OSMIŠLJENOG PLANA

Algoritam:

umnožak (n)

Ako je n = 1, onda vrati 1

inače vrati n · umnožak (n - 1)

Unesi n

Ispiši umnožak (n)

Rekurzivna funkcija

1. Rekurzivna funkcija *umnožak* ima ulazni podatak *n* koji označava najveći prirodni broj u umnošku.

2. Zaustavljanje izvršavanja funkcije.

3. Funkcija *umnožak* poziva samu sebe s izmijenjenom (manjom) ulaznom vrijednošću.

4. OSVRT NA RJEŠENJE I METODU RJEŠAVANJA

- **Stog** je struktura podataka u koju se pohranjuju varijable rekurzivne funkcije koja funkcionira na LIFO način rada.
- **LIFO** (Last In First Out) način rada znači da podataka koji je zadnji ušao prvi izlazi van.

- U trenutku kad unutar rekurzivne funkcije dođemo do rekurzivnog poziva s izmijenjenom vrijednošću argumenta, privremeno se prekida izvršavanje tekuće funkcije i vrijednost varijable sprema se na stog.
- Prvi element stoga postavlja se na dno, zatim se na njega postavlja drugi element itd.
- Postupak se ponavlja sve dok se ne dođe do osnovnog slučaja čime se završavaju rekurzivni pozivi. Tim se postupkom povećava (raste) sadržaj stoga.

Promotri kako radi ova rekurzivna funkcija za n = 5.

n = 5, poziva se funkcija *umnožak (5)*

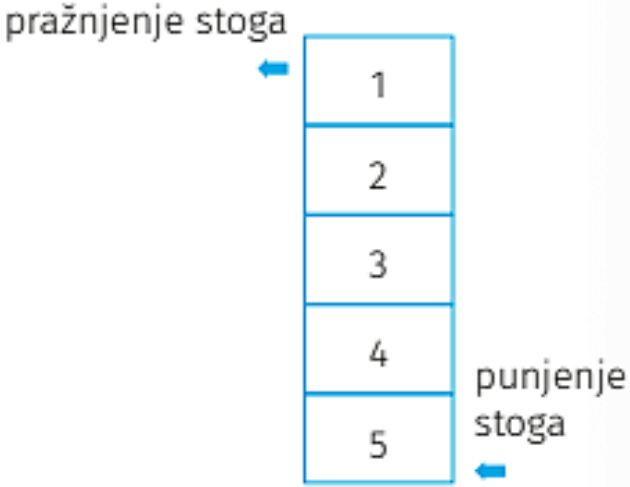
n = 5 (n ≠ 1) ➡ vrati 5 · *umnožak (4)*

n = 4 ➡ vrati 4 · umnožak (3)

n = 3 ➡ vrati 3 · umnožak (2)

n = 2 ➡ vrati 2 · umnožak (1)

n = 1 ➡ vrati 1



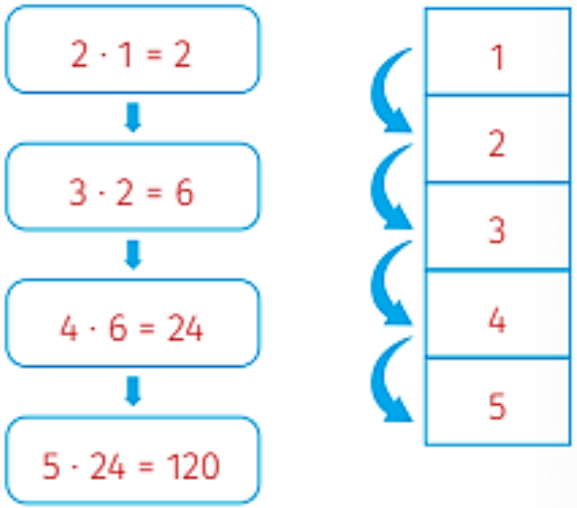
umnožak (1) = 1

umnožak (2) = 2 · umnožak (1) = 2 · 1 = 2

umnožak (3) = 3 · umnožak (2) = 3 · 2 = 6

umnožak (4) = 4 · umnožak (3) = 4 · 6 = 24

umnožak (5) = 5 · umnožak (4) = 5 · 24 = 120



ZAŠTO REKURZIJA

- Primjenom rekurzija dobijemo kraće i jednostavnije algoritme, a neke probleme bilo bi vrlo teško riješiti na neki drugi način.
- Pri izvršavanju programa rekurzivni programi su zahtjevniji i troše više memorije od iterativnih programa.
  Sporiji su u izvršavanju zato što moraju stalno spremati podatke na stog i čitati podatke s nj
- Rekurzije imaju veliku primjenu u programiranju jer se uz relativno jednostavne algoritme mogu stvarati složene strukture.

SAŽETAK

- **Samosličnost** je svojstvo objekta da sadrži dijelove u manjem mjerilu koji su slični cjelini.
- **Iteracija (petlja)** je algoritamski postupak ponavljanja naredbi određeni, konačni broj puta.
- **Rekurzija** je algoritamski postupak u kojem rekurzivna funkcija poziva samu sebe.
- **Stog** je struktura podataka u koju se pohranjuju varijable rekurzivne funkcije koja funkcionira na LIFO način rada.

PONAVLJANJE

1. Koja su dva načina rješavanja problema?
2. Kako se naziva dio rekurzivne funkcije koji dovodi do završetka funkcije?
3. Što bi se dogodilo da rekurzivna funkcija ne sadrži osnovni slučaj?
4. Što je LIFO način rada?