PANEVROPSKI UNIVERZITET APEIRON

Fakultet Informacionih tehnologija

Banja Luka

Seminarski rad iz predmeta

Principi programiranja

Naslov rada:

Dinamićko upravljanje memorijom u C jeziku

Predmetni profesor: Student:

Prof. dr Branimir Trenkić Andrej Vujanović

99-23-FITIIT1-pro-240

Banja Luka, 2025. god.

**SADRŽAJ**

[**1.** **ŠTA JE DINAMIČKO UPRAVLJANJE MEMORIJE?** 2](#_Toc188891553)

[1.1. OSNOVNA SINTAKSA ZA DINAMIČKO UPRAVLJANJE MEMORIJE 2](#_Toc188891554)

[1.2. ŠTA JE SIZEOF? 3](#_Toc188891555)

[**2.** **ŠTA JE POINTER?** 5](#_Toc188891556)

[2.1. POINTER NIZ I POINTER MATRICA 6](#_Toc188891557)

[2.2. GRAFIČKI PRIKAZ 1D I 2D DINAMIČKI ALOCIRANIH MATRICA 7](#_Toc188891558)

[2.3. KORIŠĆENJE POINTERA (SINGLE, ARRAY, 2D) U FUNKCIJI KAO ARGUMENTE 8](#_Toc188891559)

[2.4. KORIŠĆENJE POINTERA (SINGLE, ARRAY, 2D) KAO VRSTU FUNKCIJE 8](#_Toc188891560)

[**3.** **ČESTE GREŠKE** 9](#_Toc188891561)

[3.1. NEUSPJEŠNA ALOKACIJA 9](#_Toc188891562)

[3.2. CURENJE MEMORIJE 9](#_Toc188891563)

[3.3. LOGIČNE GREŠKE 9](#_Toc188891564)

[3.4. DEREFERENCIRANJE BEZ PROVJERE 9](#_Toc188891565)

[3.5. FREE FREE 10](#_Toc188891566)

[3.6. OOM GREŠKE 10](#_Toc188891567)

[3.7. FRAGMENTACIJA 10](#_Toc188891568)

[**4.** **LINKOVANE LISTE** 11](#_Toc188891569)

[4.1. PUSH I POP FUNKCIJE 13](#_Toc188891570)

[4.2. GRAFIČKI PRIKAZ PUSH I POP FUNKCIJE 15](#_Toc188891571)

[**5.** **DINAMIČKI STEK** 16](#_Toc188891572)

[5.1. ISEMPTY I TOP FUNKCIJE, GRAFIČKI PRIKAZ STEKA 16](#_Toc188891573)

[5.2. PUSH I POP FUNKCIJE 17](#_Toc188891574)

[**Izvori** 19](#_Toc188891575)

**DINAMIČKO UPRAVLJANJE MEMORIJOM U C JEZIKU**

# **ŠTA JE DINAMIČKO UPRAVLJANJE MEMORIJE?**

Dinamičko upravljanje memorije omogućava alociranje promjenjive memorije tokom pokretanja programa pomoću pointera, dok statičko upravljanje memorije alocira fiksnu memoriju tokom kompajliranja nekog programa.

Statička alokacija memorije nije prikladna kada se radi sa nekim većim programima jer ta metoda alokacije nije fleksibilna kada se rukuje sa ogromnim brojem promjenjivih. Kod takvih slučajeva koristi se dinamička alokacija memorije koja je adaptibilna potrebama većih programa i programa kod kojih varijable imaju nepredvidive memorijske zahtjeve.

## OSNOVNA SINTAKSA ZA DINAMIČKO UPRAVLJANJE MEMORIJE



Malloc alocira specifični broj bita za neki podatak, ali taj zadati prostor ne inicijalizuje pa su sve početne vrijednosti tog podatka nasumične.



Calloc alocira specifični broj bita za neki podatak, međutim za razliku od malloc-a, on ga inicijalizuje pa su sve početne vrijednosti tog podatka 0. Malloc i calloc prave pointer koji pokazuje na prostor u memoriji gdje se nalazi prvi podatak.



;

Free se koristi za dealokaciju memorije koja je zauzeta pomoću funkcija malloc i calloc. Free se koristi radi efikasnije upotrebe memorije jer bez njega program neće sam da dealocira prostor koji je zauzet sa malloc() i calloc() nakon upotrebe tih podataka.



;

Realloc() se koristi da bi dinamički promjenili veličinu alocirane memorije kod nekog podatka zbog kojeg smo već ranije u programu dinamički zauzeli memoriju, odnosno realloc se koristi u slučaju kada smo dodjelili previše ili premalo memorije nekom podatku.

Free oslobađa dinamički alociranu memoriju. Sintaksa za free glasi: free (ptr);

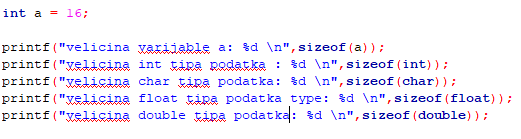
## ŠTA JE SIZEOF?

;

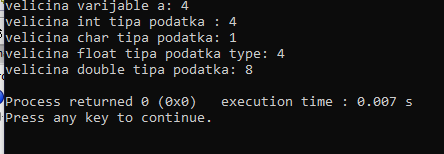


;

Sizeof operator nam pokazuje veličinu svog operanda u bajtovima. Taj operand može biti neka varijabla ili neki tip podatka. Koristimo ga u dinamičkoj alokaciji memorija da bi program znao koliko tačno bajtova da zauzme za dati podatak.



Ovaj kod će nam vratiti veličine tipa podatka int, char, float, double i varijable a (tj. int).



* 1. STATIČKA ALOKACIJA MEMORIJE

Statička alokacija memorije (*Engl Compile-Time-Memory / Static memory*), za razliku od dinamičke, vrši se tokom kompajliranja.

Osobine statičke alokacije memorije:

1. Alokacija memorije se vrši tokom kompajliranja;
2. Koristi se stek segment memorije;
3. Memorija se ne može mijenjati tokom pokretanja programa;
4. Statička alokacija memorije je brža u odnosu na dinamičku;
5. Statička alokacija memorije je manje prostorno efikasna u odnosu na dinamičku alokaciju memorije.

Mane statičke alokacije memorije:

1. Statička alokacija memorije može da koristi više memorije nego što joj je potrebno;
2. Statički alocirana memorija se ne može mjenjati tokom pokretanja programa (ne može se smanjiti, povećati ili osloboditi);
3. Mora program znati koliko mu je memorije potrebno prije pokretanja programa, da ne bi došlo do greške.

# **ŠTA JE POINTER?**

Zbog toga što tokom dinamičke alokacije memorije koristimo pointere (pokazivače), moramo da ih razumijemo i način njihovog funkcionisanja. Pointeri su varijable koje umjesto podatka sadrže neku memorijsku adresu. Sintaksa za deklarisanje praznog pointera glasi:



Nakon što deklarišemo pointer možemo da mu zadamo adresu neke varijable:



Tokom zadavanja vrijednosti pointeru koristili smo operator dereferenciranja \* i adresni operator &. Operator dereferenciranja nam služi za pristup vrijednosti koji se nalazi na nekoj adresi, dok nam adresni operator služi za dobijanje adrese nekog podatka. To znači da nakon zadavanja neke adrese pointeru mi možemo koristiti operator dereferenciranja za pristup vrijednosti te adrese.



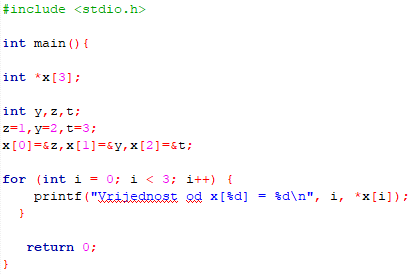


NULL pointeri su pointeri kojima je vrijednost 0. Ako imamo pointer koji ne skladišti adresu pri njegovoj deklaraciji, možemo mu dati vrijedost NULL.

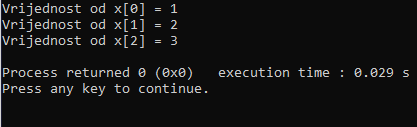


## POINTER NIZ I POINTER MATRICA

Pointeri mogu da budu i nizovi. Pointer-nizovi pokazuju na više memorijskih lokacija. Primjer Sintakse za deklarisanje pointer niza i dodjeljivanju vrijednosti tom pointer nizu:

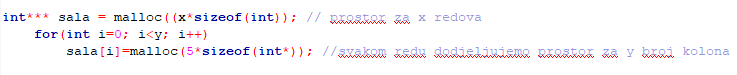


Pokretanje koda:



Možemo napraviti pointer matricu sa dinamičkom alokacijom memorije. Postoji više načina kreiranja pointer matrice:

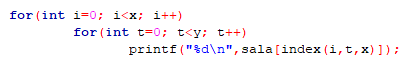
1. Predstavljanje matrice sa 1D nizom. Sintaksa za ovu metodu: 
2. Predstavljane matrice sa 2D nizom. Sintaksa za ovu metodu:



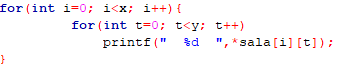
Metoda a) je lakša za inicijalizovati, ali je teže pristupiti njenim članovima nego metodi. Jedna od metoda je kreiranje funkcije za pristup članovima te funkcije.



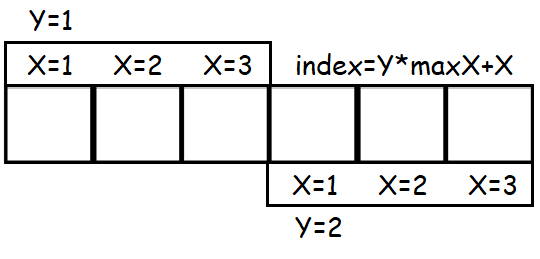
Pristupanje članovima 1D dinamički alocirane matrice.

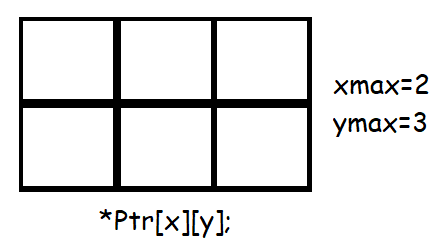


Pristupanje članovima 2D dinamičko alocirane matrice.



## GRAFIČKI PRIKAZ 1D I 2D DINAMIČKI ALOCIRANIH MATRICA

’

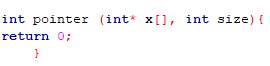


## KORIŠĆENJE POINTERA (SINGLE, ARRAY, 2D) U FUNKCIJI KAO ARGUMENTE

Single pointer:



Array Pointer:



2D Pointer:



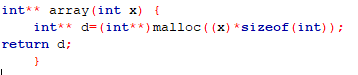
Kod korišćenja pointera sa više dimenzija kao argumena ne definiše se prva vrijednost tog višedimenzionalnog niza.

## KORIŠĆENJE POINTERA (SINGLE, ARRAY, 2D) KAO VRSTU FUNKCIJE

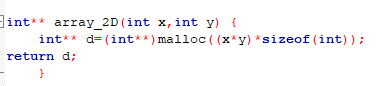
Single pointer:



Array Pointer:



2D Pointer:



# **ČESTE GREŠKE**

## NEUSPJEŠNA ALOKACIJA

Memorijska alokacija nije uvjek garantovana. U situacijama gdje alokacija memorije nije uspješna, kompajler će nam vratiti NULL pointer. Ako je alokacija uspješna, a kompajler nije provjerio da li je navedeno memorijsko mjesto prikladno za taj zahtjev (i ispostavi se da nije), program će da nam vrati nedefinisano ponašanje i doće do katastrofalnog propusta.

## CURENJE MEMORIJE

Ako nekim programom zauzimamo dosta memorije, a ne koristimo funkciju za free može doći do nagomilavanja neiskoristivih memorijskih adresa. To znači da nakon dovoljno dugog perioda nagomilavanja ovih adresa nemamo adresa gdje možemo da alociramo nove podatke

## LOGIČNE GREŠKE

Sve alokacije moraju pratiti istu rutinu. Alokacija sa malloc/calloc pa dealokacija sa free. Neuspješno praćenje ove rutine dolazi do segmentacione greške.

## DEREFERENCIRANJE BEZ PROVJERE

Ovo se dešava nakon dereferenciranja pointera bez provjeravanja da li je alokacija uspješna. Ako nije uspješna kompajler nam vraća NULL. U zavisnosti od toga gdje se koristio taj dinamički alocirani podatak, vraćanje NULL vrijednosti može dovesti do rane terminacije programa.

## FREE FREE

Još jedna česta greška je ponovo oslobađanje već oslobođenog prostora:



## OOM GREŠKE

Out Of Memory (OOM) greške se dešavaju kada nema više prostora za alociranje. Ova greška je bila češća na starijim kompjuterima i operativnim sistemima zbog njihovog ograničenog softvera i hardvera.

## FRAGMENTACIJA

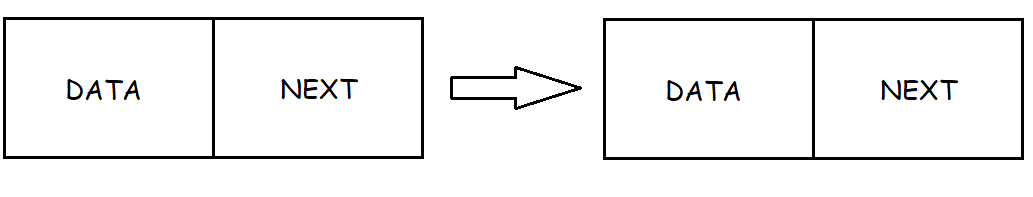
Ako je dinamičko alociranje memorije loše implementirano na većoj skali može doći do fragmentacije. Fragmentacija je greška koja se javlja kada program neefikasno koristi memoriju na računarskom sistemu tako što je ne oslobodi nakon upotrebe i uz to još i pita računarski sistem za još memorije umjesto da potpuno izkoristi memoriju koja joj je zadata. Kao rezultat te neefikasne upotrebe ta memorija je označena kao zauzeta, dok je u teoriji slobodna i ne sadrži nikakve korisne podatke. Duži periodi fragmentacije su štetni za rad računara jer postaje nemoguće da se zauzme dovoljno memorije što dovodi do znatno sporijeg rada računara.

Neki sistemi izbjegavaju dinamičku alokaciju. Program koji pomaže u izbjegavanju dinamičke alokacije je Aplov MultiFinder (Mac OS), koji unaprijed alocira onoliko RAMa za koliko aplikacija pita prilikom pokretanaja programa.

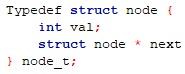
# **LINKOVANE LISTE**

Linkovana lista predstavlja vrstu liste koja je dinamički niz koji može da raste ili da se smanjuje od bilo koje tačke. Linkovane liste ne moraju imati definisanu početnu veličinu.

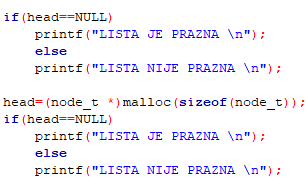
Linkovane liste se sastoje od lokalnog pointera (što znači da nije dio liste) koji pokazuje na prvi član liste, ako je taj član NULL onda je lista prazna. Prvo polje liste u slici se zove ključ, drugo polje se zove pokazivačko polje.



Definisanje linkovane liste rekurzijom:



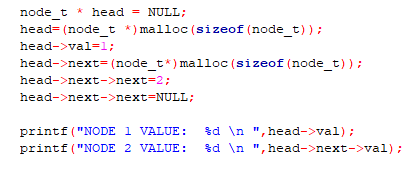
Definisanje lokalnog pokazivača koji pokazuje na prvi član liste (koji se zove glava/head):



Prvo će nam se ispisati da je lista prazna pošto smo joj zadali vrijednost NULL nakon inicijalizacije, ali nakon dinamičkog alociranja izpisaće nam suprotno pošto je malloc inicijalizovo sav zauzet prostor:



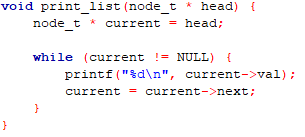
Sada zadajemo vrijednost prvoj varijabli na listi i dodajemo još jedan član listi:



Pokretanjem programa vidjećemo da vrijednost drugog člana nije inicijalizovan zato što je on zadnji član liste:



Funkcija koja nam ispisuje cijelu listu:

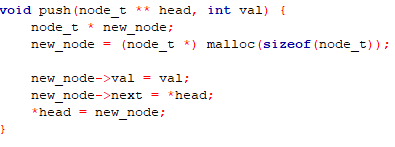




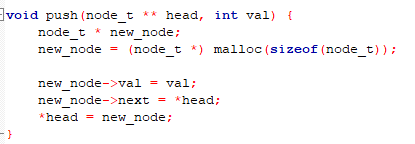


## PUSH I POP FUNKCIJE

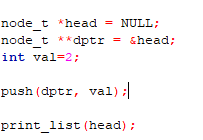
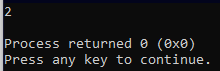
Push nam služi za dodavanje članova na početak liste, funkcija za tu operaciju izgleda ovako:



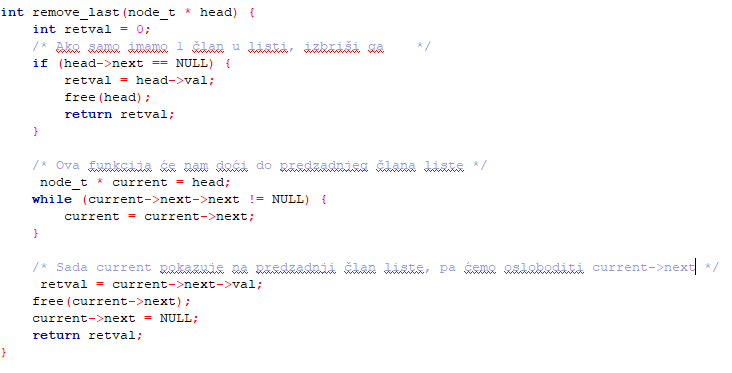
U funkciji node\_t\*\* nam je double pointer, double pointeri drže adresu nekog drugog pointera i oni se koriste ako trebamo praviti izmjene nekom pointeru u funkciji:



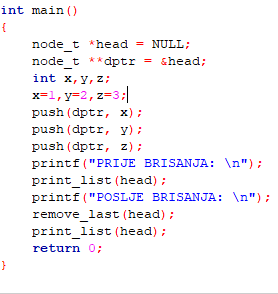
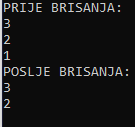
Sada iskoristimo tu funkciju da ubacimo broj 2 u listu, i da je prikažemo:

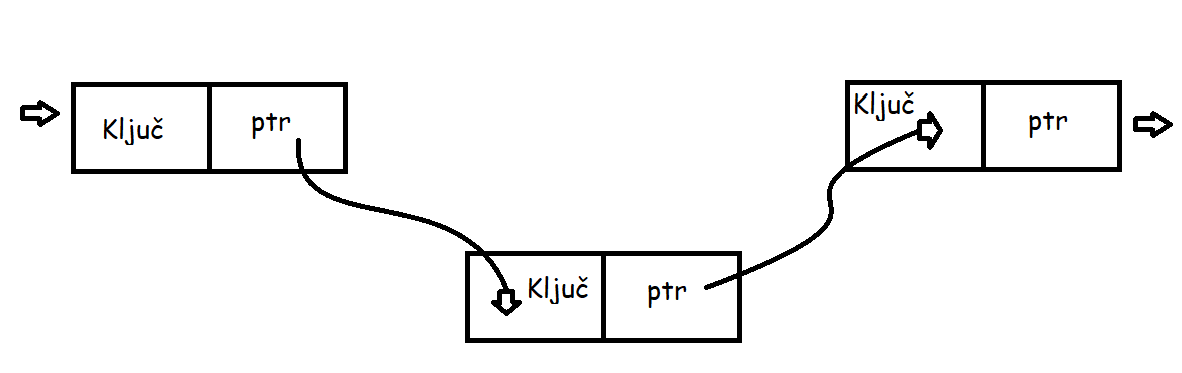
Pop nam služi za brisaje članova liste sa kraja liste. Pop funkcija treba da nam nađe predzadnji član liste, pošto zadnji nam samo služi za prikazivanje kraja liste pa njegov ključ ne sadrži nikakve podatke:



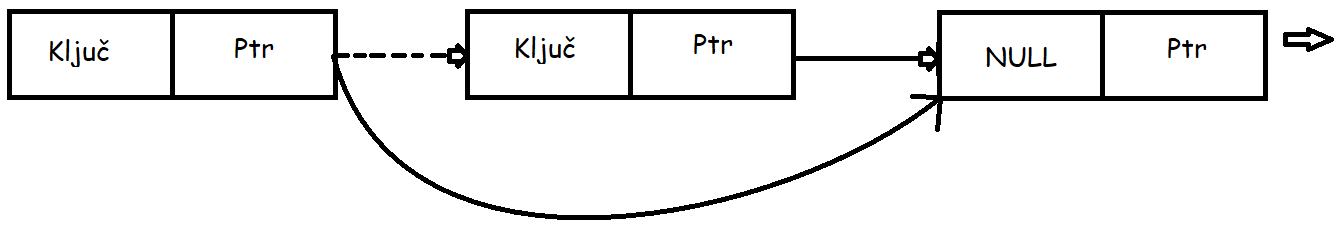
U sljedećem dijelu koda ćemo kreirati listu od 3 člana, popuniti je, ispisati je pa potom ćemo obrisati zadnji član liste:

## GRAFIČKI PRIKAZ PUSH I POP FUNKCIJE



*Grafički prikaz push funkcije*

**

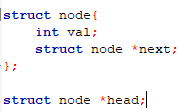
*Grafički prikaz pop funkcije*

# **DINAMIČKI STEK**

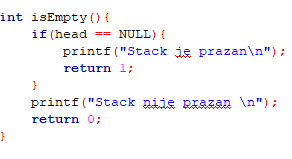
Stek je struktura podataka koja predstavlja jednu vrste liste sa određenim načinom pristupanja njenim podacima. Pristupno mjesto u steku se zove vrh (top) dok se drugi kraj steka zove dno (bottom). Ovaj način pristupanja strukturi se zove LIFO (Last in, First out). Dinamički stek je stek koji može da raste i da se smanjuje pomoću linkovane liste. Osnovne operacije sa stekom su: push (dodavanje elementa), pop (brisanje elementa), top (vraća element na vrhu), IsEmpty provjerava da li je prazan).

## ISEMPTY I TOP FUNKCIJE, GRAFIČKI PRIKAZ STEKA

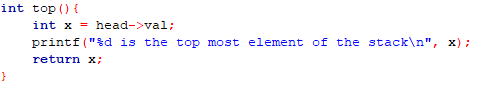
Prvo trebamo napraviti strukturu koja će nam predstavljati stek, pa onda napraviti pointer na prvi član steka prije ostalih funkcija.

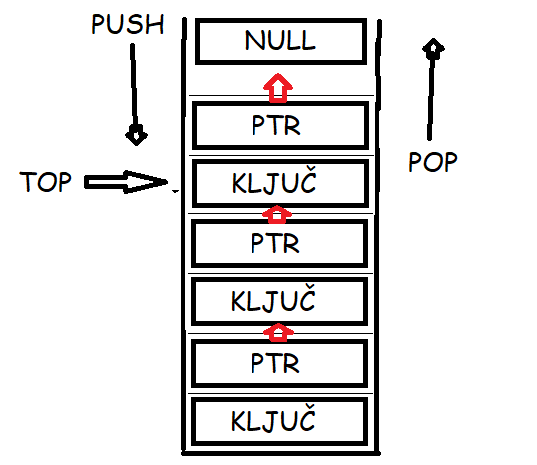


Funkcija IsEmpty će nam vratiti 1 ako je prvi član steka prazan, ali ako nije prazan vratiće nam 0.



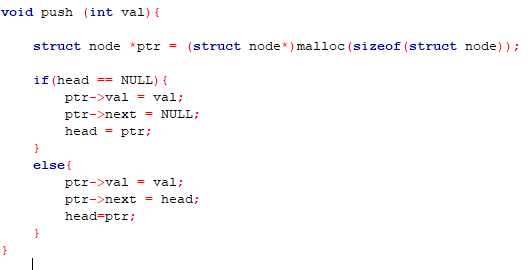
Funkcija Top će nam vraćati ključ člana koji je na vrhu steka.



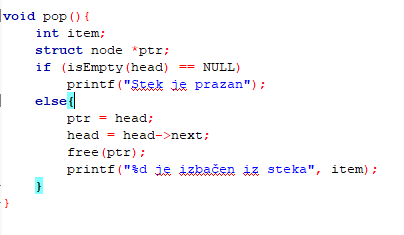
  
*Grafički prikaz dinamičkog steka*

## PUSH I POP FUNKCIJE

Push funkcija će alocirati mjesto koje zauzima veličinu jednog člana steka (ptr) , pa će onda u njega ubaciti vrijednost val. Naš glavni stek head će uzeti tu vrijednost val, nakon čega ćemo pomjeriti glavni stek head za jedno mjesto.



Pop funkcija će prvo provjeriti da li je stek prazan koristeći funkciju IsEmpty, pa će onda inicijalizovati pointer koji iste vrste kao glavni stek (Ptr). Taj pointer će pokazivati na član steka koji se nalazi na vrhu, nakon čega ćemo pomjeriti naš glavni stek head za jedan član. Na kraju svega ovoga oslobađamo pointer Ptr.



# **Izvori**

* Branimir M. Trenkić: Algoritmi i strukture podataka - prvo izdanje (2023)
* <https://www.geeksforgeeks.org>/
* [https://www.sanfoundry.com/](https://www.sanfoundry.com/c-tutorials-common-errors-while-memory-being-allocated-dynamically/#:~:text=The%20most%20common%20error%20is,it%20causes%20abnormal%20program%20termination)
* [https://www.eskimo.com/](https://www.eskimo.com/~scs/cclass/notes/sx11.html)
* C Standard Manual (2007.9.7) https://www.open- std.org/jtc1/sc22/wg14/www/docs/n1256.pdf
* https://en.wikipedia.org/wiki/C\_dynamic\_memory\_allocation