**X[Y] ⬄ \*(X+Y)**

**X – adresa**

**Y – integer vrijednost**

# Zadatak 1. – Adrese elemenata niza

Napravite program slijedeće navedene zahtjeve:

a)

* deklarišete niz A tipa *integer* dužine 5 i inicijalizujete ga sa vrijednostima {201, 202, 203, 204, 205}
* ispišite uzastopno adrese elemenata niza

|  |
| --- |
| *Pomoć:*   * adresa prvog elementa: &A[0] * adresa drugog element: &A[1] … itd. * Zaključujemo da se adrese uzastopnih elemenata razlikuju za 4 bajta (to je veličina podatka tipa *int*) |

b) prepravite prethodni program:

* ispišite adresu niza sa naredbom cout <<&A;

|  |
| --- |
| Info: **&A** predstavlja adresu prvog člana niza isto kao **cout <<&A[0];** |

* ispišite adresu niza sa naredbom cout << A;

|  |
| --- |
| * + **naziv niza** (A) predstavlja adresu prvog člana niza   + &A[0] == &A == A |

b) prepravite prethodni program:

* deklarišite pokazivače p1, p2, p3, p4, p5 na *int*
* inicijalizujte pokazivačima vrijednost, tako da:
  + p1 pokazuje na prvi element niza A,
  + p2 pokazuje na drugi element niza A,
  + itd.
* ispišite vrijednosti niza dereferenciranjem pokazivača p1, p2, .., p5

|  |
| --- |
| **Info** |
| Inkrementiranjem adrese niza x ili korištenje izraza „x + 1“ uvećava se adresa x za veličinu jednog element niza.   |  |  | | --- | --- | | 1:  2:  3:  4:  5:  6:  7:  8: | void main()  {  int A [5];  cout << (&A[0])+1 << endl; // adresa drugog elementa  cout <<&A[1] << endl; // adresa drugog elementa  cout << A+1 << endl; // adresa drugog elementa  //... | |

# Zadatak 2. – Adrese elemenata niza

Deklarišite niz A tipa *int* veličine 5. Ispišite adrese elemenata niza koristeći operator '+' bez upotrebe adresnog operatora '&'. Koristite *for*-petlju.

Zatim, ispišite razliku između adrese elementa sa indeksom 4 i adrese elementa sa indeksom 1 koristeći operator '-' i operator '&'.

# Zadatak 3. – Dereferenciranje elemenata

Pristup vrijednosti elementa niza sa indeksom 1 možete postići naredbom (cout << A[1]). Ispišite vrijednost elementa niza sa indeksom 1 dereferenciranjem njegove adrese. Dereferenciranje možete postići operatorom '**\***'.

# Zadatak 4. – Pokazivač na niz (pokazivač na pokazivač)

Deklarišite tri varijable (a, b, c) i inicijaluzujte ih

* deklarišite **niz pokazivača** B veličine 100
* neka prva tri elementa niza pokazuju na varijable a, b, c
* ispišite vrijednosti varijabli a, b, c koristeći prva tri elementa niza

**Pitanje: Kojeg tipa varijable je „B“ !**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| *Odgovor:*   * 'B' predstavlja adresu prvog elementa niza B (to znači da bi B trebao biti neki pokazivač) * 'B[0]' predstavlja prvi pokazivač koji pokazuje na varijablu a, tj. 'B[0]' je, ipak, pokazivač koji čuva adresu, to znači: sada B[0] nije vrijednost, pa je treba dereferencirati * \*(B[0]) predstavlja konkretnu vrijednost (podatak tipa *int*, broj 201) * pogledajmo ponovo izraz '\*(B[0])' , vidimo da je dva puta vršeno dereferenciranje kako bi smo dobili konkretnu vrijednost (podatak tipa *int*, broj 201) * zaključujemo: 'B' je podatak tipa ***pokazivač na pokazivač*** * pogledajte tabelu koja se odnosi na rješenje ovog zadatka!  |  |  |  |  | | --- | --- | --- | --- | |  | **Tip podatka** | **Vrijednosti** | **Opis** | | B | pokazivač na pokazivač na integer | neka adresa | ime niza | | B[1] | Pokazivač na integer | neka adresa | vrijednosti drugog elementa, tj. adresa varijable b | | \***(**B[1]**)** | integer | 202 | vrijednost varijable b | | \***(**\*B**)** | integer | 201 | vrijednost varijable a | | \***(**\*(B+1)**)** | integer | 202 | vrijednost varijable b | |

# Zadatak 5. – Pokazivač na elemenat niza



int B[5];

int\* p1;

Pitanje:

*Šta je rezultat sljedećeg programa?*

|  |  |
| --- | --- |
| 13:  13'  14:  14':  15:  15': | p1 = B+2; // ili  /\* sljedeće linije koda obavljaju identične operacije:  - 13 i 13'  - 14 i 14'  - 15 i 15'  \*/  p1 = B; p1++; p1++;  cout << \*(p1+0); // ili  cout << p1[0];  cout << \*(p1+2); // ili  cout << p1[2]; |

# Zadatak 7. – Niz pokazivača

a)

* Deklarišite niz A tipa *integer* veličine 99 i inicijalizujte elemente na vrijednosti 301, 302, 303, .., 399
* Deklarišite niz B veličine 33, tipa *pokazivač* na *integer*
* Inicijalizujte elemente niza B tako da pokazivači pokazuju na svaki treći element niza A, npr:
  + prvi element niza B treba da pokazuje na prvi element niza A
  + drugi element niza B treba da pokazuje na četvrti element niza A, …
* Ispišite sve vrijednosti *integera* na koji niz B pokazuje (dereferenciranjem elemenata niza B)

b) Prepravite rješenje tako da dereferencirate pomoću operatora '\*' umjesto operatora '[]'

# Zadatak 8.– Jednodimenzionalni niz kao parametar funkcije

1. Implementirajte funkciju voiducitaj(int\* p, intv)
2. Implementirajte funkciju voidispis(int\* p, intv)
3. Implementirajte funkciju main:

* deklarišite konstanu v = 10
* deklarišite niz N veličine v.
* pozovite funkciju ucitaj
* pozovite funkciju ispis

# Zadatak 9 – Dinamička alokacija niza

Prepravite prethodni program tako da korisnik odrediti veličinu niza.

# Zadatak 10 – Dinamička alokacija niza

**Napišite program u kojem će:**

* Korisnik odrediti veličinu **x**.
* Alocirati dinamičke nizove A veličine **x**
* Alocirati dinamičke nizove B veličine **x**
* Korisnik treba unijeti elemente nizova A i B.
  + koristite funkciju **ucitaj** iz prethodnog zadatka.
* Alocirati dinamičke nizove C veličine **x**
* Sabrati elemente nizova A i B, te rezultat smjestiti u niz C
  + koristite funkciji **saberi**.
* Ispisati elemente niza C
  + koristite funkcije **ispis** iz prethodnog zadatka.

# Zadatak12 – Dinamička alokacija niza

**Napišite program u kojem ćete:**

* Kreirati dinamički niz od onoliko cjelobrojnih elemenata koliko želi korisnik;
* Omogućiti inicijalizaciju članova niza unosom s tastature;
* Utvrditi koliko elemenata niza je manje od 0;
* Kreirati novi niz čiji će elementi biti elementi prvog niza koji zadovoljavaju prethodni uvjet (manji su od 0);
* Koristite funkcije.

# Zadatak 13 – Niz pokazivača

a)



* deklarišite nizove R1, R2 i R3 veličine 4 tipa *int*
  + koristite konstantu
* deklarišite niz M tipa *pokazivač na int* veličine 3
  + koristite konstantu
* neka prvi element niza M (prvi pokazivač) pokazuje na niz R1, drugi element na R2 i treći na R3, kao na slici:

b)

- učitajte sve elemente koristeći pokazivač M

c)

- ispišite sve elemente koristeći pokazivač M

# Zadatak 14 – Dvodimenzionalni niz kao parametar funkcije

Prepravite prethodni program tako da koristite sljedeće funkcije

void ispis(int\*\* X, int r, int k)

void ucitaj(int\*\* X, int r, int k)

# Zadatak 15 – Dvodimenzionalni niz kao parametar funkcije

Prepravite prethodni program tako da se statički nizovi R1, R2 i R3 zamjene sa dinamičkom alokacijom.

# Zadatak 16 – Dvodimenzionalni niz kao parametar funkcije

Prepravite prethodni program tako da se nizovi R1, R2 i R3 zamjene sa dinamičkom alokacijom i to bez korištenja dodatnih pokazivača, tj. za alokaciju svih nizova dozvoljeno je koristiti samo niz pokazivača **M**.

# Zadatak 17 – Dvodimenzionalni niz kao parametar funkcije

Prepravite prethodni program tako da se niz pokazivača M alocira dinamički.

# Zadatak 18 – Dvodimenzionalni niz kao parametar funkcije

Prepravite prethodni program tako da korisnik može odrediti broj kolona i broj redova matrice M.

# Zadatak 19 – Dvodimenzionalni niz kao parametar funkcije

U prethodni program dodajte sljedeće funkcije:

int Izracunaj\_sumu(int\*\* X, int r, int k)

float Izracunaj\_prosjek(int\*\* X, int r,int k)

# Zadatak 20 – Provjera znanja\*\*

Napravite program slijedeći navedene zahtjeve. Svi zahtjevi su povezani i predstavljaju jedan cijeli program. Preporuka je da od koraka ***g*** svaku promjenu pokazivača p1 i p2 skicirate.

1. deklarišite niz **N** tipa integer dužine 15 u statičkoj memoriji
2. deklarišite pokazivače **p1** i **p2** na integer(e) u jednoj liniji koda
3. alocirajte niz u dinamičkom memoriji dužine 19, i neka **p1** pokazuje taj niz
4. alocirajte niz u dinamičkom memoriji dužine 12, i neka **p2** pokazuje taj niz
5. zamijenite pokazivače, neka **p1** pokazuje na niz dužine 12, a **p2** na niz dužine 19
6. reciklirajte niz dužine 12 i učinite da njegov pokazivač (**p1**) bude null
7. neka **p1** pokazuje na elemenat niza sa indeksom 5 na koji pokazuje **p2**, kao na slici:



1. dodijelite neku (proizvoljnu) vrijednost elementu sa indeksom 5 dereferenciranjem **p1***(pomoću operatora '\*')*
2. dodijelite neku vrijednost elementu sa indeksom 5 dereferenciranjem njegove izračunate adrese izrazom **p2+5** *(pomoću operatora '\*')*
3. dodijelite neku vrijednost elementu sa indeksom 2 dereferenciranjem njegove izračunate adrese preko **p2** *(pomoću operatora '\*')*
4. dodijelite neku vrijednost elementu sa indeksom 2 dereferenciranjem njegove izračunate adrese preko **p1** *(pomoću operatora '\*')*
5. ispišite vrijednost elementa sa indeksom 5 dereferenciranjem pokazivača **p1** *(pomoću operatora '[ ]')*
6. ispišite vrijednost elementa sa indeksom 5 dereferenciranjem pokazivača **p2** *(pomoću operatora '[ ]')*
7. ispišite vrijednost elementa sa indeksom 2 dereferenciranjem pokazivača **p2** *(pomoću operatora '[ ]')*
8. ispišite vrijednost elementa sa indeksom 2 dereferenciranjem pokazivača ***p1*** *(pomoću operatora '[ ]')*
9. reciklirajte dinamički niz dužine 19
10. neka **p1** pokazuje na prvi elemenat niza **N** (elemenat sa indeksom 0)
11. dodijelite neku vrijednost elementu sa indeksom 2 dereferenciranjem njegove izračunate adrese preko **p1** *(pomoću operatora '\*')*
12. dodijelite neku vrijednost elementu sa indeksom 2 dereferenciranjem njegove izračunate adrese preko **N** *(pomoću operatora '\*')*
13. ispišite vrijednost elementa sa indeksom 2 dereferenciranjem pokazivača **p1** *(pomoću operatora '[ ]')*
14. ispišite vrijednost elementa sa indeksom 2 dereferenciranjem pokazivača **N** *(pomoću operatora '[ ]')*

*Pomoć:*

|  |  |
| --- | --- |
| h) | * p1 pokazuje na elemenat sa indeksom 5; * potrebno je samo dereferencirati adresu p1 da bi pristupili elementu sa indeksom 5; |
| k) | * p1 predstavlja adresu elementa sa indeksom 5; * dok (p1-3) predstavlja adresu elementa sa indeksom 2; * da bi pristupili elementu sa indeksom 2 potrebno je samo dereferencirati (p1-3); |

# Zadatak 21 – Provjera znanja

* kreirajte dinamički niz objekata tipa radnik od onoliko članova koliko želi korisnik; struct radnik

{

int ib;

char ime[10];

char prezime[10]; double dohodak[12]; int staz;

char spol[10];

};

* kreirati funkciju za unos obilježja za sve objekte: void unos(radnik \*, int);
* kreirati funkciju za ispis obilježja jednog objekta: void ispis(radnik \*);
* kreirati funkciju koja će prebrojati koliko radnika su žene:

int prebroji\_zene(radnik \*, int);

* kreirati novi niz u dinamičkoj memoriji;
* kreirati funkciju u kojoj ćete kopirati sve žene u novonapravljeni niz kopirati sve žene:

void kopiraj\_zene(radnik \*, radnik \*, int);

* kreirati funkciju koja će pronaći i vratiti radnika s najdužim prezimenom:

radnik \* najduze\_ime(radnik \*, int);

* kreirati funkciju koja će vršiti pretragu radnika po imenu, te ispisati sva obilježja radnika za koje je utvrđeno da imaju traženo ime:

void pretraga\_po\_imenu(radnik \*, char \*, int);

* kreirati funkciju koja će izračunati godišnje prihode za izabranog radnika:

double prihodi(radnik \*, int, int); radnika se bira preko IB-a radnika, a prihodi se računaju tako da sve zbroje dohoci za 12 mjeseci + 6% za svaku godinu staža;