# 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  //... | |

|  |
| --- |
| **Rješenje** |
| int A[] = { 201, 202, 203, 204, 205 };  cout << &A[0] << endl;  cout << &A[1] << endl;  cout << &A[2] << endl;  cout << &A[3] << endl;  cout << &A[4] << endl;  cout << A + 0 << endl;  cout << A + 1 << endl;  cout << A + 2 << endl;  cout << A + 3 << endl;  cout << A + 4 << endl;  int\* p1 = A + 0;  int\* p2 = A + 1;  int\* p3 = A + 2;  int\* p4 = A + 3;  int\* p5 = A + 4;  cout << \*p1 << endl;  cout << \*p2 << endl;  cout << \*p3 << endl;  cout << \*p4 << endl;  cout << \*p5 << endl; |

# 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 '&'.

|  |
| --- |
| **Rješenje** |
| void main()  {  const int n=5;  int A[n];  for (int i=0; i<n; i++)  cout << "adresa elementa " << i << ": " << (A + i) << endl;  cout << "Razlika između elementa 1 i elementa 4 je " << &A[4] - &A[1];  } |

# 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 '**\***'.

|  |
| --- |
| **Rješenje** |
| cout << "Drugi clan: " << **\***(&A[0]+1) << endl;  cout << "Drugi clan: " << **\***(&A[1]) << endl;  cout << "Drugi clan: " << A[1] << endl;  cout << "Drugi clan: " << **\***(A+1) << endl; |

|  |
| --- |
| *Zaključak:*  Adresni operator '&' i operatori dereferenciranja su međusobno inverzni.  U operatore defereniciranja spadaju:   * operator '**\***' * operator srednje zagrade '**[ ]**' * operator '**->**' za objekte, (o ovome će biti riječi u narednim poglavljima)   Primjeri 1 – linija kôda br. 4:   * 'A' predstavlja adresu (adresu prvog elementa) * dok 'A[1]' predstavlja neku vrijednost (podatak tipa *int*)   Primjeri 2 – linija kôda br. 6:   * 'A' predstavlja adresu (adresu prvog elementa) * dok '**\***(A+1)' predstavlja neku vrijednost (podatak tipa *int*)   Primjeri 3 – linija kôda br. 3:   * 'A' predstavlja adresu (adresu prvog elementa) * 'A[1]' predstavlja neku vrijednost (ovdje *int*) * '&A[1]' predstavlja opet adresu (adresu drugog elementa) * '\*(&A[1])' predstavlja vrijednost (podatak tipa *int*) |

# 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 | |

**X[Y] ⬄ \*(X+Y)**

**X adresa**

**Y integer**

|  |
| --- |
| **Rješenje** |
| void main()  {  int a = 201, b = 202, c = 203;  int\* B[100] = {&a, &b, &c};  cout << "prva vrijednost: " << \*(B[0]) << endl;  cout << "druga vrijednost: " << \*(B[1]) << endl;  cout << "treca vrijednost: " << \*(B[2]) << endl;  cout << "prva vrijednost: " << \*\*(B+0) << endl;  cout << "druga vrijednost: " << \*\*(B+1) << endl;  cout << "treca vrijednost: " << \*\*(B+2) << endl;  } |

# Zadatak 5. – Pokazivač na elemenat niza



int B[5];

int\* p1;

**X[Y] ⬄ \*(X+Y)**

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 '[]'

|  |  |
| --- | --- |
| *Rješenje - prvi način* | |
| 1:  2:  3:  4:  5:  6:  7:  8:  9:  10:  11:  12:  13:  14:  15:  16:  17: | #include <iostream>  using namespace std;  void main()  {  int A[99];  for (int i=0; i<**=**98; i++)  A[i] = 301+i;  int\* B[33];  for (i=0; i<**=**32; i++)  B[i] = &(A[3\*i]); //ili  B[i] = (A + 3\*i);  for (i=0; i<**=**32; i++)  cout << \*(B[i]) << endl;  cout << \*\*(B+i) << endl;  } |

|  |  |
| --- | --- |
| *(ili jednostavnije pomoću zajedničke konstantne kao dužina niza:)* | |
| 1:  2:  3:  4:  5:  6:  7:  8:  9:  10:  11:  12:  13:  14:  15:  16: | void main()  {  const int x = 33;  int A[3\*x];  for (int i=0; i < 3\*x; i++)  A[i] = 301+i;  int\* B[x];  for (int i=0; i<=x-1; i++)  B[i] = &(A[3\*i]);  for (int i=0; i<=x-1; i++)  cout << \*(B[i]) << endl;  } |

# Zadatak 8. – Jednodimenzionalni niz kao parametar funkcije

1. Implementirajte funkciju void ucitaj(int\* p, int v)
2. Implementirajte funkciju void ispis(int\* p, int v)
3. Implementirajte funkciju main:

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

|  |  |
| --- | --- |
| **Rješenje** | |
| 1:  2:  3:  4:  5:  6:  7:  8:  9:  10:  11:  12:  13:  14:  15:  16:  17:  18:  19:  20:  21:  22:  23:  24:  25:  26:  27: | #include <iostream>  using namespace std;  void ispis(int\* p, int v)  {  for (int i = 0; i<v; i++)  cout << i + 1 << ": " << p[i] << endl;  }  void ucitaj(int\* p, int v)  {  cout << "Unesite elemente niza: \n";  for (int i = 0; i<v; i++)  {  cout << i + 1 << ": ";  cin >> p[i];  }  }  void main()  {  const int v = 10;  int N[v];  ucitaj(N, v);  cout << "==========\n";  ispis(N, v);  } |

# Zadatak 9 – Dinamička alokacija niza

Prepravite prethodni program tako da korisnik odrediti veličinu niza.

|  |
| --- |
| **Rješenje** |
| void main()  {  int v;  cout << "Unesite veličinu niza: ";  cin >> v;  int\* N = new int[v];  ucitaj(N, v);  cout << "==========\n";  ispis(N, v);  } |

# 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.

|  |  |
| --- | --- |
| **Rješenje** | |
| 1:  2:  3:  4:  5:  6:  7:  8:  9:  10:  11:  12:  13:  14:  15:  16:  17:  18:  19:  20:  21:  22:  23:  24:  25:  26:  27:  28:  29:  30:  31:  32:  33:  34:  35:  36:  37:  38:  39: | #include <iostream>  using namespace std;  void ispis(int\* p, int x)  {  for (int i = 0; i<x; i++)  cout << i + 1 << ": " << p[i] << endl;  }  void ucitaj(int\* p, int x)  {  cout << "\nUnesite elemente niza: \n";  for (int i = 0; i<x; i++)  {  cout << i + 1 << ": ";  cin >> p[i];  }  }  void saberi(int\* a, int\* b, int\* c, int x)  {  for (int i = 0; i<x; i++)  c[i] = a[i] + b[i];  }  void main()  {  int x;  cout << "Koliko elemenata zelis unijeti: ";  cin >> x;  int\* A = new int[x];  int\* B = new int[x];  int\* C = new int[x];      ucitaj(A, x);  ucitaj(B, x);  saberi(A, B, C, x);  cout << "\n===suma===\n";  ispis(C, x);  } |

# Zadatak 12 – Dinamička alokacija niza

***(preuzeto iz PR II - V6 - 2015/2016)***

**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.

|  |
| --- |
| **Rješenje** |
| #include <iostream>  #include <iomanip>  using namespace std;  int unos(int \*, int);//funkcija vraca vrijednost jer ce osim unosa elemenata i prebrojati koliko elemenata je manje od 0  void inicijalizacija\_niza(int \*, int\*, int);//funkcija smjesta negativne elemente iz prvog niza u drugi  void ispis(int \*, int);  void main()  {  int velicina1, velicina2;  cout << "Unesite zeljenu velicinu niza" << endl;  cin >> velicina1;  int \*Pokazivac1 = new int[velicina1];//alokacija prvog niza    cout << "Unesite elemente niza" << endl;  velicina2 = unos(Pokazivac1, velicina1);    int \*Pokazivac2 = new int[velicina2]; //alokacija drugog niza    inicijalizacija\_niza(Pokazivac1, Pokazivac2, velicina1);    cout << "Prvi niz ima " << velicina2 << " elemenata manjih od nula, a to su " << endl;  ispis(Pokazivac2, velicina2);  cout << endl;  delete[] Pokazivac1; //obavezno dealocirajte memoriju  delete[] Pokazivac2;  Pokazivac1 = NULL;  Pokazivac2 = NULL;  system("pause>0");  }  int unos(int \*pok, int vel)//funkciju za unos elemenata prvog niza smo iskoristili da prebrojimo koliko elemenata zadovoljava trazeni uvjet (manji od 0)  {  int brojac = 0;  for (int i = 0; i<vel; i++)  {  cin >> pok[i];  if (pok[i]<0)  brojac++;  }  return brojac;  }  void inicijalizacija\_niza(int \*pok1, int\* pok2, int vel)  {  int j = 0;  for (int i = 0; i<vel; i++)  {  if (pok1[i]<0)  {  pok2[j] = pok1[i];  j++;  }  }  }  void ispis(int \* pok, int vel)  {  for (int i = 0; i<vel; i++)  cout << setw(4) << pok[i];  } |

# Zadatak 13 – Pokazivač na niz

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

|  |  |
| --- | --- |
| **Rješenje** | |
| 1:  2:  3:  4:  5:  6:  7:  8:  9:  10:  11:  12:  13:  14:  15:  16:  17:  18:  19:  20:  21:  22:  23:  24:  25:  26:  27:  28: | #include <iostream>  using namespace std;  void main()  {  const int k = 4;  const int r = 3;  int R1[k], R2[k], R3[k];  int\* M[r] = { R1, R2, R3 };    cout << "\n=====citanje=====\n";  for (int i = 0; i < r; i++)  {  cout << "Unesite " << k << " elementa za red " << i << ">>>> ";  for (int j = 0; j < k; j++)  cin >> M[i][j];  cout << endl;  }  cout << "\n=====ispis=====\n";  for (int i = 0; i<r; i++)  {  for (int j = 0; j<k; j++)  {  cout << M[i][j] << " | ";  }  cout << endl;  }  } |

# 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)

|  |
| --- |
| **Rješenje** |
| #include <iostream>  using namespace std;  void ispis(int\*\* X, int r, int k)  {  for (int i = 0; i<r; i++)  {  for (int j = 0; j<k; j++)  {  cout << X[i][j] << " | ";  }  cout << endl;  }  }  void ucitaj(int\*\* X, int r, int k)  {  for (int i = 0; i < r; i++)  {  cout << "Unesite " << k << " elementa za red br " << i << ": ";  for (int j = 0; j < k; j++)  cin >> X[i][j];  cout << endl;  }  }  void main()  {  const int k = 4;  const int r = 3;  int R1[k], R2[k], R3[k];  int\* M[r] = { R1, R2, R3 };    cout << "\n=====citanje=====\n";  ucitaj(M, r, k);  cout << "\n=====ispis=====\n";  ispis(M, r, k);  } |

# Zadatak 15 – Dvodimenzionalni niz kao parametar funkcije

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

|  |
| --- |
| **Rješenje** |
| void main()  {  int k = 4;  int\* R1 = new int[k]; //ovo je potrebno na kraju programa dealocirati  int\* R2 = new int[k]; //ovo je potrebno na kraju programa dealocirati  int\* R3 = new int[k]; //ovo je potrebno na kraju programa dealocirati  const int r = 3;  int\* M[r] = {R1, R2, R3};  cout << "\n=====citanje=====\n";  ucitaj(M, r, k);  cout << "\n=====ispis=====\n";  ispis(M, r, k);  delete[] R1;  delete[] R2;  delete[] R3;  } |

# 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**.

|  |
| --- |
| **Rješenje** |
| void main()  {  int k = 4;  const int r = 3;  int\* M[r];  M[0] = new int[k]; //ovo je potrebno na kraju programa dealocirati  M[1] = new int[k]; //ovo je potrebno na kraju programa dealocirati  M[2] = new int[k]; //ovo je potrebno na kraju programa dealocirati  cout << "\n=====citanje=====\n";  ucitaj(M, r, k);  cout << "\n=====ispis=====\n";  ispis(M, r, k);  delete[] M[0];  delete[] M[1];  delete[] M[2];  } |

# Zadatak 17 – Dvodimenzionalni niz kao parametar funkcije

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

|  |
| --- |
| **Rješenje** |
| void main()  {  int k = 4;  int r = 3;  int\*\* M = new int\*[r];  M[0] = new int[k];  M[1] = new int[k];  M[2] = new int[k];  cout << "\n=====citanje=====\n";  ucitaj(M, r, k);  cout << "\n=====ispis=====\n";  ispis(M, r, k);  delete[] M[0];  delete[] M[1];  delete[] M[2];  delete[] M;  } |

# Zadatak 18 – Dvodimenzionalni niz kao parametar funkcije

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

|  |
| --- |
| **Rješenje** |
| void main()  {  int k, r;  cout << "Unesi broj redova: ";  cin >> r;  cout << "Unesi broj kolona: ";  cin >> k;  int\*\* M = new int\*[r];  for (int i = 0; i<r; i++)  M[i] = new int[k];  cout << "\n=====citanje=====\n";  ucitaj(M, r, k);  cout << "\n=====ispis=====\n";  ispis(M, r, k);  for (int i = 0; i<r; i++)  delete[] M[i];  delete[] 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)

|  |
| --- |
| int suma(int\*\* X, int r, int k)  {  int s = 0;  for (int i = 0; i < r; i++)  {  for (int j = 0; j < k; j++)  s+=X[i][j];  }  return s;  }  float prosjek(int\*\* X, int r, int k)  {  return suma(X, r, k) / float(r\*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); |

|  |  |  |
| --- | --- | --- |
| **Rješenje** | | |
| 1:  2:  3:  4:  5:  6:  7:  8:  9:  10:  11:  12:  13:  14:  15:  16:  17:  18:  19:  20:  21:  22:  23:  24:  25:  26:  27:  28:  29:  30:  31:  32:  33:  34:  35:  36:  37:  38:  39:  40:  41:  42:  43:  44:  45:  46:  47:  48:  49:  50:  51:  52: | #include <iostream>  using namespace std;  void main()  {  //a) deklarišite niz N tipa integer dužine 15 u statičkoj memoriji  int N[15];  //b) deklarišite pokazivače p1 i p2 na integer(e) u jednoj liniji koda  int \*p1, \*p2;  //c) alocirajte niz u dinamičkom memoriji dužine 19, i neka p1 pokazuje taj niz  p1 = new int[19];  //d) alocirajte niz u dinamičkom memoriji dužine 12, i neka p2 pokazuje taj niz  p2 = new int[12];  //e) zamijenite pokazivače, neka p1 pokazuje na niz dužine 12, a p2 na niz dužine 19  int\* temp;  temp = p1;  p1 = p2;  p2 = temp;  //f) reciklirajte niz dužine 12 i učinite da njegov pokazivač (p1) bude ,,bezopasan*''*  delete [] p1;  p1 = NULL;  //g) neka p1 pokazuje na elemenat niza sa indeksom 5 na koji pokazuje p2  p1 = &p2[5];  //h) dodijelite neku vrijednost elementu sa indeksom 5 dereferenciranjem p1  \*p1 = 124;  //i) dodijelite neku vrijednost elementu sa indeksom 5 deref. njegove izračunate adrese izrazom p2+5  \*(p2+5) = 125;  //j) dodijelite neku vrijednost elementu sa indeksom 2 deref. njegove izračunate adrese preko p2  \*(p2+2) = 126;  //k) dodijelite neku vrijednost elementu sa indeksom 2 deref. njegove izračunate adrese preko p1  \*(p1-3) = 127;  //l) ispišite vrijednost elementa sa indeksom 5 koristeći operator '[ ]' na pokazivaču p1  cout << p1[0] << endl;  //m) ispišite vrijednost elementa sa indeksom 5 koristeći operator '[ ]' na pokazivaču p2  cout << p2[5] << endl;  //n) ispišite vrijednost elementa sa indeksom 2 koristeći operator '[ ]' na pokazivaču p2  cout << p2[2] << endl;  //o) ispišite vrijednost elementa sa indeksom 2 koristeći operator '[ ]' na pokazivaču p1  cout << p1[-3] << endl;  //p) reciklirajte dinamički niz dužine 19  delete [] p2;  //q) neka p1 pokazuje na prvi elemenat niza N (elemenat sa indeksom 0)  p1 = &N[0]; // ili p1 = N;  //r) dodijelite neku vrijednost elementu sa indeksom 2 deref. njegove izračunate adrese preko p1  \*(p1+2) = 128;  //s) dodijelite neku vrijednost elementu sa indeksom 2 deref. njegove izračunate adrese preko N  \*(N+2) = 129;  //t) ispišite vrijednost elementa sa indeksom 2 koristeći operator '[ ]' na pokazivaču p1  cout << p1[2] << endl;  //u) ispišite vrijednost elementa sa indeksom 2 koristeći operator '[ ]' na (pokazivaču) N  cout << N[2] << endl;  } | 1  1  1  1  2  2  2  0  1  1  1  1  1 |

# Zadatak 21 – Provjera znanja

***(preuzeto iz PR II - V6 - 2015/2016)***

* 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;

|  |
| --- |
| **Rješenje** |
| #include<iostream>  #include<iomanip>  using namespace std;  struct radnik  {  int ib;  char ime[10];  char prezime[10];  double dohodak[12];  int staz;  char spol[10];  };  void unos(radnik \*, int);//funkcija za unos obiljezja za sve objekte  void ispis(radnik \*);//funkcija za ispis obiljezja jednog objekta  int prebroji\_zene(radnik \*, int); //funkcija koja prebrojava koliko radnika je zenskog spola  void kopiraj\_zene(radnik \*, radnik \*, int);  radnik \* najduze\_prezime(radnik \*, int);  void pretraga\_po\_imenu(radnik \*, char \*, int);  double prihodi(radnik \*, int, int);  void main()  {  int brElemenata, brElemenata2, IB;  char nizIME[10];  cout << "Koliko elementa ce imati niz radnika" << endl;  cin >> brElemenata; cin.ignore();  radnik \* nizRadnika = new radnik[brElemenata];//kreiranje dinamickog niza objekata  cout << "Unesite obiljezja svih objekata" << endl<<endl;  unos(nizRadnika, brElemenata); //unos obiljezja za sve radnike  cout << "Ispis obiljezja objekata" << endl << endl;    for (int i = 0; i < brElemenata; i++)  {  ispis(&nizRadnika[i]);//funkcija ispisuje obiljezja samo jednog radnika  cout << endl;  }  brElemenata2 = prebroji\_zene(nizRadnika, brElemenata);//vrijednost koju vrati funkcija prebroji\_zene je velicina novog niza  radnik \*nizZena = new radnik[brElemenata2];  kopiraj\_zene(nizRadnika, nizZena, brElemenata);//funkcija kopira sve zene u novokreirani niz  cout << "Ispis obiljezja objekata novog niza - zene" << endl << endl;  for (int i = 0; i < brElemenata2; i++) //ispis obiljezja objekata novog niza - zena  ispis(&nizZena[i]);  cout << "Ispis obiljezja radnika s najduzim prezimenom " << endl << endl;  ispis(najduze\_prezime(nizRadnika, brElemenata));    cout << "Pretraga po imenu " << endl << endl << "Unesite ime za pretragu " << endl << endl;  cin.getline(nizIME, 10);    pretraga\_po\_imenu(nizRadnika, nizIME, brElemenata);  cout << "Izaberite radnika ciji godisnji dohodak zelite izracunati " << endl << endl;  cout << "Unesite IB " << endl;  cin >> IB;  cout << "Godisnji dohodak radnika s ib: " << IB << " iznosi " << prihodi(nizRadnika, brElemenata, IB) << " KM" << endl;  system("PAUSE>0");  }  void unos(radnik \* niz, int vel)  {  for (int i = 0; i < vel; i++)  {  cout << "Unesite obiljezja " << i + 1 << " objekta " << endl;  cout << "IB: "; cin >> niz[i].ib; cin.ignore();  cout << "Ime: ";  cin.getline(niz[i].ime, 10);  //funkcija cin.getline omogucava unos niza karaktera specificirane duzine  cout << "Prezime: ";  cin.getline(niz[i].prezime, 12);    for (int j = 0; j < 12; j++)  {  cout << "Mjesec " << setw(3) << j + 1 << ": ";  cin >> niz[i].dohodak[j];  }  cout << "Staz: ";  cin >> niz[i].staz;  cin.ignore();  cout << "Spol: ";  cin.getline(niz[i].spol, 10);  cin.ignore();  cout << endl;  }  }  void ispis(radnik \* niz)  {  cout << "IB: " << niz->ib << endl;  cout << "Ime: " << niz->ime << endl;  cout << "Prezime: " << niz->prezime << endl;  for (int j = 0; j < 12; j++)  cout << "Mjesec " << setw(3) << j + 1 << ": " << setw(3) << niz->dohodak[j] << endl;    cout << "Staz: " << niz->staz << endl;  cout << "Spol: " << niz->spol << endl << endl;  }  int prebroji\_zene(radnik \* niz, int vel)  {  int brojac = 0;  for (int i = 0; i < vel; i++)  {  if (strcmp(niz[i].spol, "zenski") == 0)  //funkcija strcmp usporedjuje dva niza karaktera i vraca 0 ako su nizovi isti  brojac++;  }  return brojac;  }  void kopiraj\_zene(radnik \* nR, radnik \* nZ, int vel)  {  int j = 0;  for (int i = 0; i < vel; i++)  {  if (strcmp(nR[i].spol, "zenski") == 0)  nZ[j] = nR[i];  }  }  radnik \* najduze\_prezime(radnik \* nizRadnika, int vel)  {  int brojslova = 0, pom = 0;  radnik \* pokazivac = 0;  for (int i = 0; i < vel; i++)  {  pom = strlen(nizRadnika[i].prezime); //funkcija strlen vraca duzinu niza karaktera  if (pom > brojslova)  {  brojslova = pom;  pokazivac = nizRadnika + i;  }  }  return pokazivac;  }  void pretraga\_po\_imenu(radnik \* nizRadnika, char \*IME, int vel)  {  for (int i = 0; i < vel; i++)  {  if (strcmp(nizRadnika[i].ime, IME) == 0)  {  cout << "Postoji radnik/ci s tim imenom " << endl;  ispis(&nizRadnika[i]);  }  }  }  double prihodi(radnik \* nizRadnika, int vel, int IB)  {  double placa = 0;  for (int i = 0; i < vel; i++)  {  if (nizRadnika[i].ib == IB)  for (int j = 0; j < 12; j++)  placa += nizRadnika[i].dohodak[j] + nizRadnika[i].dohodak[j] \* 0.06;  }  return placa;  } |

# Zadatak 22 – Provjera znanja

a)

Prepravite program sljedeći program tako da :

* svi objekti budu se trebaju alocirati u dinamičkoj memoriji
* sve funkcije trebaju primati adrese pokazivača na objekat umjesto vrijednost objekta
* sve funkcije koje vraća reference ili objekte trebaju vraćati adresu na objekat

Link na program (ovo je rješenje od Zadaci03\_Pokazivac – Z4)

|  |
| --- |
| #include <iostream>  using namespace std;  // stuktura Datum  struct Datum {  int d;  int m;  int g;  };  void info(Datum date) {  // dd/mm/gg  cout << date.d << "/" << date.m << "/" << date.g << ". ";  }  Datum ucitaj\_datum()  {  Datum x;  cout << "unesite dan, mjesec i godinu: ";  cin >> x.d;  cin >> x.m;  cin >> x.g;  return x;  }  // struktura Osoba  struct Osoba {  int id;  char ime[15];  };  void info(Osoba o) {  cout << "Ime osobe je: " << o.ime << " id= " << o.id << ". ";  }  // Studenti  const int max\_s = 5;  Osoba studenti[max\_s] = {  { 1, "Marko" },  { 2, "Darko" },  { 3, "Zarko" },  { 4, "Adil" },  { 5, "Faruk" }  };  Osoba odaberite\_osobu(Osoba niz[], int z)  {  int a;  cout << "Spisak osoba: " << endl;  for (int i = 0; i < z; i++)  {  cout << i << "Ime osobe: " << niz[i].ime << ", ID: " << niz[i].id << endl;  }  cout << "Unesite RB: ";  cin >> a;  Osoba x = niz[a];  return x;  }  Osoba o1 = { 534, "Nina B." };  Osoba o2 = { 435, "Jasmin A."};  Osoba o3 = { 256, "Denis M." };  Osoba o4 = { 325, "Emina J." };  Osoba o5 = { 914, "Zanin V." };  const int max\_i = 5;  Osoba ispitivaci[max\_i] = { o1, o2, o3, o4, o5,};  struct PrijavaZaIspit  {  Osoba student;  Datum datum\_prijave;  Datum datum\_odjave;  int ocjena;  };  const int max\_p = 100;  struct Ispit {  Osoba ispitivac;  Datum datum\_ispita;  char predmet[10];  int brojac\_prijavljenih;  PrijavaZaIspit prijave[max\_p];  };  Ispit ucitaj\_ispit()  {  Ispit x;  cout << "Unesite predmet " << endl;  cin >> x.predmet;  cout << "Odaberite ispitivaca" << endl;  x.ispitivac = odaberite\_osobu(ispitivaci, max\_i);  cout << "Unesite datum ispita" << endl;  x.datum\_ispita = ucitaj\_datum();    x.brojac\_prijavljenih = 0;  return x;  }  void info(PrijavaZaIspit p)  {  cout << "Student: ";  info(p.student);  cout << " Datum prijave: ";  info(p.datum\_prijave);  cout << " Datum odjave: ";  info(p.datum\_odjave);  cout << endl;  }  void prikazi\_ispite(Ispit ispiti[], int max)  {  for (int i = 0; i < max; i++)  {  cout << i +1 << ". ";  info(ispiti[i].datum\_ispita);  info(ispiti[i].ispitivac);  cout << "Predmet -> " << ispiti[i].predmet ;  cout << ", \tBroj prijavljenih: " << ispiti[i].brojac\_prijavljenih << endl;  }  }  Ispit& odaberite\_ispit(Ispit ispiti[], int max)  {  prikazi\_ispite(ispiti, max);  cout << "Unesite RB ispita: ";  int x;  cin >> x;  return ispiti[x - 1];  }  void main()  {  const int max\_ispiti = 10;  Ispit ispiti[max\_ispiti];  int brojac\_ispita = 0;  int x;  do  {  cout << "1. Ispiti - prikazi sve" << endl;  cout << "2. Ispiti - dodaj novi (zakazi novi ispit)" << endl;  cout << "3. Prijavljeni za ispit: prikazi sve (za ispit x)" << endl;  cout << "4. Prijavljeni za ispit: dodaj novu prijavu (za ispit x)" << endl;  cout << "5. Evidentiraj sve ocjene (za ispit x)" << endl;  cout << "0. EXIT" << endl;  cin >> x;  if (x == 1)  {  prikazi\_ispite(ispiti, brojac\_ispita);  }  if (x == 2)  {  Ispit i = ucitaj\_ispit();  ispiti[brojac\_ispita] = i;  brojac\_ispita++;  }  if (x == 3)  {  Ispit i = odaberite\_ispit(ispiti, brojac\_ispita);  for (int j = 0; j<i.brojac\_prijavljenih; j++)  {  PrijavaZaIspit& p = i.prijave[j];  info(p);  }  }  if (x == 4)  {  Ispit& i = odaberite\_ispit(ispiti, brojac\_ispita);  PrijavaZaIspit novaP;  novaP.student = odaberite\_osobu(studenti, max\_s);  novaP.datum\_prijave = ucitaj\_datum();  i.prijave[i.brojac\_prijavljenih++] = novaP;  }  if (x == 5)  {  Ispit& i = odaberite\_ispit(ispiti, brojac\_ispita);  for (int j = 0; j<i.brojac\_prijavljenih; j++)  {  info(i.prijave[j].student);  cout << ", Unesite ocjenu sa ispita: ";  cin >> i.prijave[j].ocjena;  }  }  } while (x != 0);  } |

|  |
| --- |
| **Rješenje** |
|  |

b)

Prepravite prethodni program tako da koristite niz pokazivača.

# Zadatak 23 – Provjera znanja

Prepravite prethodni program tako da umjesto običnih pokazivača koristite pametni pokazivač shared\_ptr

Primjer definisana pokazivača

shared\_ptr<Datum> date (new Datum);

Primjer pristupa članovima objekta na koji pokazuje pokazivač

date->d

|  |
| --- |
| #include <iostream>  #include <memory>  using namespace std;  //----------struktura DATUM-------------  struct Datum  {  int d;  int m;  int g;  };  void info(shared\_ptr<Datum> date)  {  if (date == nullptr)  return;  cout << date->d << "." << date->m << "." << date->g;  }  shared\_ptr<Datum> ucitaj\_datum()  {  shared\_ptr<Datum> date(new Datum);  cout << "Unesite dan, mjesec, godinu:" << endl;  cin >> date->d >> date->m >> date->g;  return date;  }  //----------struktura OSOBA-------------  struct Osoba  {  char ime[10];  int id;  };  void info(shared\_ptr<Osoba> o)  {  if (o == nullptr)  return;  cout << "Id: \t" << o->id << ", Ime: " << o->ime;  }  shared\_ptr<Osoba> odaberite\_osobu(shared\_ptr<Osoba> osobe[], int max)  {  for (int i = 0; i<max; i++)  {  info(osobe[i]);  cout << endl;  }  do  {  cout << "Unesite ID osobe: ";  int x;  cin >> x;  for (int i = 0; i < max; i++)  {  if (osobe[i]->id == x)  return osobe[i];  }  cout << "Neispravan Id" << endl;  } while (true);  }  Osoba\* getOsoba(char\* ime, int id)  {  Osoba\* o = new Osoba;  strncpy\_s(o->ime, ime, 10);  o->id = id;  return o;  }  const int max\_s = 10;  shared\_ptr<Osoba> studenti[] = {  shared\_ptr<Osoba>(getOsoba("Student J", 101)),  shared\_ptr<Osoba>(getOsoba("Student I", 102)),  shared\_ptr<Osoba>(getOsoba("Student H", 103)),  shared\_ptr<Osoba>(getOsoba("Student G", 104)),  shared\_ptr<Osoba>(getOsoba("Student F", 105)),  shared\_ptr<Osoba>(getOsoba("Student E", 106)),  shared\_ptr<Osoba>(getOsoba("Student D", 107)),  shared\_ptr<Osoba>(getOsoba("Student C", 108)),  shared\_ptr<Osoba>(getOsoba("Student B", 109)),  shared\_ptr<Osoba>(getOsoba("Student A", 110))  };  const int max\_i = 5;  shared\_ptr<Osoba> ispitivaci[max\_i] = {  shared\_ptr<Osoba>(getOsoba("Nina B.", 534)),  shared\_ptr<Osoba>(getOsoba("Jasmin A.", 435)),  shared\_ptr<Osoba>(getOsoba("Denis M.", 256)),  shared\_ptr<Osoba>(getOsoba("Emina J.", 325)),  shared\_ptr<Osoba>(getOsoba("Zanin V.", 914)),  };  struct PrijavaZaIspit  {  shared\_ptr<Osoba> student;  shared\_ptr<Datum> datum\_prijave;  shared\_ptr<Datum> datum\_odjave;  int ocjena;  };  const int max\_p = 100;  struct Ispit  {  shared\_ptr<Osoba> ispitivac;  shared\_ptr<Datum> datum\_ispita;  int brojac\_prijavljenih;  char predmet[10];  shared\_ptr<PrijavaZaIspit>\* prijave = new shared\_ptr<PrijavaZaIspit>[max\_p]; //umjesto PrijavaZaIspit prijave[max\_p];  };  shared\_ptr<Ispit> dodaj\_ispit()  {  shared\_ptr<Ispit> i(new Ispit);  i->brojac\_prijavljenih = 0;  //ili  //i->brojac\_prijavljenih = 0;  cout << "Unesite naziv predmeta (bez razmaka)" << endl;  cin >> i->predmet;  cout << "Unesite datum ispita: ";  i->datum\_ispita = ucitaj\_datum();  cout << "Ispitivac: ";  i->ispitivac = odaberite\_osobu(ispitivaci, max\_i);  return i;  }  void info(shared\_ptr<Ispit> i)  {  if (i == nullptr)  return;  cout << "Predmet: " << i->predmet;  cout << ", \tDatum ispita: ";  info(i->datum\_ispita);  cout << ", \tIspitivac: ";  info(i->ispitivac);  cout << ", \tBroj prijavljenih: " << i->brojac\_prijavljenih << endl;  }  shared\_ptr<Ispit> odaberite\_ispit(shared\_ptr<Ispit> ispiti[], int max)  {  for (int i = 0; i<max; i++)  {  cout << i + 1 << ": ";  info(ispiti[i]);  }  cout << "Unesite RB ispita: ";  int x;  cin >> x;  return ispiti[x - 1];  }  void main()  {  const int max\_ispiti = 10;  shared\_ptr<Ispit>\* ispiti = new shared\_ptr<Ispit>[max\_ispiti];  int brojac\_ispita = 0;  int x;  do  {  cout << "1. Ispiti - prikazi sve" << endl;  cout << "2. Ispiti - dodaj novi (zakazi novi ispit)" << endl;  cout << "3. Prijavljeni za ispit: prikazi sve (za ispit x)" << endl;  cout << "4. Prijavljeni za ispit: dodaj novu prijavu (za ispit x)" << endl;  cout << "5. Evidentiraj sve ocjene (za ispit x)" << endl;  cout << "0. EXIT" << endl;  cin >> x;  if (x == 1)  {  for (int i = 0; i<brojac\_ispita; i++)  {  info(ispiti[i]);  }  }  if (x == 2)  {  shared\_ptr<Ispit> i = dodaj\_ispit();  ispiti[brojac\_ispita] = i;  brojac\_ispita++;  }  if (x == 3)  {  shared\_ptr<Ispit> i = odaberite\_ispit(ispiti, brojac\_ispita);  for (int j = 0; j<i->brojac\_prijavljenih; j++)  {  shared\_ptr<Osoba> s = i->prijave[j]->student;  shared\_ptr<Datum> dP = i->prijave[j]->datum\_prijave;  shared\_ptr<Datum> dO = i->prijave[j]->datum\_odjave;  info(s);  cout << " Datum prijave: ";  info(dP);  cout << " Datum odjave: ";  info(dO);  cout << endl;  }  }  if (x == 4)  {  shared\_ptr<Ispit> i = odaberite\_ispit(ispiti, brojac\_ispita);  shared\_ptr<PrijavaZaIspit> novaP(new PrijavaZaIspit);  novaP->student = odaberite\_osobu(studenti, max\_s);  novaP->datum\_prijave = ucitaj\_datum();  novaP->datum\_odjave = nullptr;  i->prijave[i->brojac\_prijavljenih++] = novaP;  }  if (x == 5)  {  shared\_ptr<Ispit> i = odaberite\_ispit(ispiti, brojac\_ispita);  for (int j = 0; j<i->brojac\_prijavljenih; j++)  {  info(i->prijave[j]->student);  cout << "Unesite ocjenu sa ispita: ";  cin >> i->prijave[j]->ocjena;  }  }  } while (x != 0);  } |