**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  //... | |

|  |
| --- |
| **Rješenje** |
|  |

# 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** |
|  |

# 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 | |

|  |
| --- |
| **Rješenje** |
|  |

# 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 '[]'

|  |
| --- |
| *Rješenje* |

# 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** | |
|  |  |

# Zadatak 9 – Dinamička alokacija niza

Prepravite prethodni program tako da korisnik odrediti veličinu niza.

|  |
| --- |
| **Rješenje** |
|  |

# 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** | |
|  |  |

# 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** |
|  |

# 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

|  |  |
| --- | --- |
| **Rješenje** | |
|  |  |

# 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** |
|  |

# Zadatak 15 – Dvodimenzionalni niz kao parametar funkcije

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

|  |
| --- |
| **Rješenje** |
|  |

# 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** |
|  |

# Zadatak 17 – Dvodimenzionalni niz kao parametar funkcije

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

|  |
| --- |
| **Rješenje** |
|  |

# Zadatak 18 – Dvodimenzionalni niz kao parametar funkcije

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

|  |
| --- |
| **Rješenje** |
|  |

# 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); |

|  |  |  |
| --- | --- | --- |
| **Rješenje** | | |
|  |  |  |

# 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** |
|  |

# 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)

|  |
| --- |
| <http://redmine.fit.ba:91/redmine/attachments/download/2117/DL_2017_03_10_z4_ispiti.txt> |
| #include <iostream>  using namespace std;  //----------struktura DATUM-------------  struct Datum  {  int d;  int m;  int g;  };  void info(Datum date)  {  cout << date.d << "." << date.m << "." << date.g;  }  Datum ucitaj\_datum()  {  Datum date;  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(Osoba o)  {  cout << "Id: \t" << o.id << ", Ime: " << o.ime;  }  Osoba odaberite\_osobu(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);  }  const int max\_s = 10;  Osoba studenti[max\_s] = {  { "Student J", 101 },  { "Student I", 102 },  { "Student H", 103 },  { "Student G", 104 },  { "Student F", 105 },  { "Student E", 106 },  { "Student D", 107 },  { "Student C", 108 },  { "Student B", 109 },  { "Student A", 110 },  };  const int max\_i = 5;  Osoba ispitivaci[max\_i] = {  { "Nina B.", 534 },  { "Jasmin A.", 435 },  { "Denis M.", 256 },  { "Emina J.", 325 },  { "Zanin V.", 914 },  };  struct PrijavaZaIspit  {  Osoba student;  Datum datum\_prijave;  Datum datum\_odjave;  int ocjena;  };  const int max\_p = 100;  struct Ispit  {  Osoba ispitivac;  Datum datum\_ispita;  int brojac\_prijavljenih;  char predmet[10];  PrijavaZaIspit prijave[max\_p];  };  Ispit dodaj\_ispit()  {  Ispit i;  //zadatak: ucitati podatke  i.brojac\_prijavljenih = 0;  cout << "Unesite naziv predmeta (bez razmaka)" << endl;  cin >> i.predmet;  cout << "Unesite datum ispita: ";  i.datum\_ispita = ucitaj\_datum();  i.datum\_ispita.d = 0;  i.datum\_ispita.m = 0;  i.datum\_ispita.g = 0;  cout << "Ispitivac: ";  i.ispitivac = odaberite\_osobu(ispitivaci, max\_i);  return i;  }  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 info(Ispit i)  {  cout << "Predmet: " << i.predmet;  cout << ", \tDatum ispita: ";  info(i.datum\_ispita);  cout << ", \tIspitivac: ";  info(i.ispitivac);  cout << ", \tBroj prijavljenih: " << i.brojac\_prijavljenih << endl;  }  Ispit& odaberite\_ispit(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;  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)  {  for (int i = 0; i<brojac\_ispita; i++)  {  info(ispiti[i]);  }  }  if (x == 2)  {  Ispit i = dodaj\_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;  Osoba(char\* ime, int id)  {  strcpy(this->ime, ime);  this->id = 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);  }  const int max\_s = 10;  shared\_ptr<Osoba> studenti[] = {  shared\_ptr<Osoba>( new Osoba("Student J", 101 )),  shared\_ptr<Osoba>( new Osoba("Student I", 102 )),  shared\_ptr<Osoba>( new Osoba("Student H", 103 )),  shared\_ptr<Osoba>( new Osoba("Student G", 104 )),  shared\_ptr<Osoba>( new Osoba("Student F", 105 )),  shared\_ptr<Osoba>( new Osoba("Student E", 106 )),  shared\_ptr<Osoba>( new Osoba("Student D", 107 )),  shared\_ptr<Osoba>( new Osoba("Student C", 108 )),  shared\_ptr<Osoba>( new Osoba("Student B", 109 )),  shared\_ptr<Osoba>( new Osoba("Student A", 110 ))  };  const int max\_i = 5;  shared\_ptr<Osoba> ispitivaci[max\_i] = {  shared\_ptr<Osoba>( new Osoba("Nina B.", 534 )),  shared\_ptr<Osoba>( new Osoba("Jasmin A.", 435) ),  shared\_ptr<Osoba>( new Osoba("Denis M.", 256 )),  shared\_ptr<Osoba>( new Osoba("Emina J.", 325 )),  shared\_ptr<Osoba>( new Osoba("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);  } |