# Zadatak 1

*Dovršiti program*

|  |
| --- |
| #include <iostream>  #include <fstream>  using namespace std;  char \* crt = "\n========================================\n";  enum NacinStudiranja { REDOVAN, DL };  enum GodinaStudija { PRVA = 1, DRUGA, TRECA, CETVRTA };  enum Odgovori { TACNO, NETACNO };  const int brojPitanja = 5;  struct Student  {  int \_brojIndeksa;  char \_imePrezime[30];  bool \_radioTest;  NacinStudiranja \_nacin;  GodinaStudija \_godina;  Odgovori \_odgovori[brojPitanja];  };  void PrikaziPitanje(int rbPitanja) {  if (rbPitanja == 1)  cout << crt << "Enumeracije i unije su sinonimi ?" << crt;  else if (rbPitanja == 2)  cout << crt << "Moguce je kreirati anonimnu uniju ?" << crt;  else if (rbPitanja == 3)  cout << crt << "Svi clanovi unije koriste istu memorijsku lokaciju?" << crt;  else if (rbPitanja == 4)  cout << crt << "Svi clanovi strukture koriste istu memorijsku lokaciju ? " << crt;  else  cout << crt << "Unija moze biti deklarisana unutar strukture ?" << crt;  }  Odgovori GetTacanOdgovor(int rbPitanja) {  if (rbPitanja == 1)  return NETACNO;  if (rbPitanja == 2)  return TACNO;  if (rbPitanja == 3)  return TACNO;  if (rbPitanja == 4)  return NETACNO;  return TACNO;  }  int GetUspjehStudenta(Student & s) {  int brojacTacnih = 0;  // usporediti odgovore studenta iz niza "s.odgovori[]" sa tačnim odgovorima preko funkcije "GetTacanOdgovor"  return brojacTacnih;  }  void PokreniTest(Student & student) {  // prikazati sva pitanja preko funkcije "PrikaziPitanje"  // sa konzole učitati odgovore i evidentirati ih u niz "student.\_odgovori[]"  for (size\_t i = 0; i < brojPitanja; i++)  {  PrikaziPitanje(i);  cout << "Unesite odgovor: 1-tacno,2-netacno. " << endl;  //student.\_odgovori[i] = ...;  }  student.\_radioTest = true;  }  void UnosPodataka(Student niz[], int max)  {  int unos;  for (int i = 0; i<max; i++)  {  cout << crt << "\t::STUDENT " << i + 1 << "::" << crt;  cout << "Unesite ime i prezime: ";  cin.getline(niz[i].\_imePrezime, 30);  cout << "Unesite broj indeksa: ";  cin >> niz[i].\_brojIndeksa;  cout << "Nacin studiranja (1.Redovan, 2.DL): ";  //... cin + enum?  cout << "Nacin godinu studija (1,2,3,4): ";  //... cin + enum?  cin.ignore();  niz[i].\_radioTest = false;//tek smo kreirali studenta  //tako da je sigurno da test nije radjen  }  }  int PrikaziMeni() {  int izbor;  do  {  cout << crt << "\t::MENI::" << crt;  cout << "1. Pokreni test" << endl;  cout << "2. Prikazi rezultate" << endl;  cout << "3. Kraj rada" << crt;  cout << "Unesite izbor: ";  cin >> izbor;  } while (izbor<1 || izbor>3);  return izbor;  }  int PrikaziStudente(Student niz[], int max)  {  int rb;  do  {  for (int i = 0; i<max; i++)  cout << i + 1 << ". " << niz[i].\_imePrezime << endl;  cout << crt << "Unesite redni broj studenta: ";  cin >> rb;  } while (rb<1 || rb>max);  return rb - 1;  }  void main()  {  int max, izbor;  cout << "Unesite broj studenata: ";  cin >> max;  cin.ignore();  Student \* niz = new Student[max];  UnosPodataka(niz, max);  system("cls");  do  {  izbor = PrikaziMeni();  if (izbor == 3)  break;  int s = PrikaziStudente(niz, max);  if (izbor == 1)  {  //pokreni test za student s  }  if (izbor == 2)  {  //prikaži odgovore i bodove za student s  }  system("pause>0");  system("cls");  } while (1);  delete[] niz;  niz = NULL;  system("pause");  } |

|  |
| --- |
| **Rješenje** |
| #include <iostream>  #include <fstream>  using namespace std;  char \* crt = "\n========================================\n";  enum NacinStudiranja { REDOVAN, DL };  enum GodinaStudija { PRVA = 1, DRUGA, TRECA, CETVRTA };  enum Odgovori { TACNO, NETACNO };  const int brojPitanja = 5;  struct Student  {  int \_brojIndeksa;  char \_imePrezime[30];  bool \_radioTest;  NacinStudiranja \_nacin;  GodinaStudija \_godina;  Odgovori \_odgovori[brojPitanja];  };  void PrikaziPitanje(int rbPitanja) {  if (rbPitanja == 1)  cout << crt << "Enumeracije i unije su sinonimi ?" << crt;  else if (rbPitanja == 2)  cout << crt << "Moguce je kreirati anonimnu uniju ?" << crt;  else if (rbPitanja == 3)  cout << crt << "Svi clanovi unije koriste istu memorijsku lokaciju?" << crt;  else if (rbPitanja == 4)  cout << crt << "Svi clanovi strukture koriste istu memorijsku lokaciju ? " << crt;  else  cout << crt << "Unija moze biti deklarisana unutar strukture ?" << crt;  }  Odgovori GetTacanOdgovor(int rbPitanja) {  if (rbPitanja == 1)  return NETACNO;  if (rbPitanja == 2)  return TACNO;  if (rbPitanja == 3)  return TACNO;  if (rbPitanja == 4)  return NETACNO;  return TACNO;  }  int GetUspjehStudenta(Student & s) {  int brojacTacnih = 0;  // usporediti odgovore studenta iz niza "s.odgovori[]" sa tačnim odgovorima preko funkcije "GetTacanOdgovor"  for (int i = 0; i < brojPitanja; i++) {  if (s.\_odgovori[i] == GetTacanOdgovor(i+1))  {  brojacTacnih++;  }  }  return brojacTacnih;  }  void PokreniTest(Student & student) {  // prikazati sva pitanja preko funkcije "PrikaziPitanje"  // sa konzole učitati odgovore i evidentirati ih u niz "student.\_odgovori[]"  for (size\_t i = 0; i < brojPitanja; i++)  {  PrikaziPitanje(i+1);  cout << "Unesite odgovor: 1-tacno,2-netacno. " << endl;  int x;  cin >> x;  x--;  student.\_odgovori[i] =(Odgovori)x ;  }  student.\_radioTest = true;  }  void UnosPodataka(Student niz[], int max)  {  int unos;  for (int i = 0; i<max; i++)  {  cout << crt << "\t::STUDENT " << i + 1 << "::" << crt;  cout << "Unesite ime i prezime: ";  cin.getline(niz[i].\_imePrezime, 30);  cout << "Unesite broj indeksa: ";  cin >> niz[i].\_brojIndeksa;  cout << "Nacin studiranja (1.Redovan, 2.DL): ";  //... cin + enum?  int unos1;  cin >> unos1;  NacinStudiranja n = (NacinStudiranja)unos1;  niz[i].\_nacin = n;  cout << "Nacin godinu studija (1,2,3,4): ";  //... cin + enum?  int unos2;  cin >> unos2;  GodinaStudija g = (GodinaStudija)unos2;  niz[i].\_godina = g;  cin.ignore();  niz[i].\_radioTest = false;//tek smo kreirali studenta  //tako da je sigurno da test nije radjen  }  }  int PrikaziMeni() {  int izbor;  do  {  cout << crt << "\t::MENI::" << crt;  cout << "1. Pokreni test" << endl;  cout << "2. Prikazi rezultate" << endl;  cout << "3. Kraj rada" << crt;  cout << "Unesite izbor: ";  cin >> izbor;  } while (izbor<1 || izbor>3);  return izbor;  }  int PrikaziStudente(Student niz[], int max)  {  int rb;  do  {  for (int i = 0; i<max; i++)  cout << i + 1 << ". " << niz[i].\_imePrezime << endl;  cout << crt << "Unesite redni broj studenta: ";  cin >> rb;  } while (rb<1 || rb>max);  return rb - 1;  }  void main()  {  int max, izbor;  cout << "Unesite broj studenata: ";  cin >> max;  cin.ignore();  Student \* niz = new Student[max];  UnosPodataka(niz, max);  system("cls");  do  {  izbor = PrikaziMeni();  if (izbor == 3)  break;  int s = PrikaziStudente(niz, max);  if (izbor == 1)  {  //pokreni test za student s  Student& st = niz[s];  PokreniTest(st);  //niz[s] = st;  }  if (izbor == 2)  {  //prikaži bodove za student s  Student& st = niz[s];  int t = GetUspjehStudenta(st);  int p = 100\* t / brojPitanja;  cout << "Uspjeh je " << p << endl;  }  system("pause>0");  system("cls");  } while (1);  delete[] niz;  niz = NULL;  system("pause");  } |

Zadataci 1, 2 i 3 su preuzeti iz DL materijala (PR-II V)

# Zadatak 2

*U prethodni zadatak implementirati funkciju*

|  |
| --- |
| void Pretraga(Student niz[], int max)  {  //definisati funkciju(e) koja(e) ce omoguciti pretragu po:  //1. godini studija  //2. nacinu studiranja  //3. rezultatima ostvarenim na testu  //Nove funkcionalnosti dodati u funkciju PrikaziMeni  } |

|  |
| --- |
| **Rješenje** |
| #include <iostream>  #include <fstream>  using namespace std;  char \* crt = "\n========================================\n";  enum NacinStudiranja { REDOVAN, DL };  enum GodinaStudija { PRVA = 1, DRUGA, TRECA, CETVRTA };  enum Odgovori { TACNO, NETACNO };  const int brojPitanja = 5;  struct Student  {  int \_brojIndeksa;  char \_imePrezime[30];  bool \_radioTest;  NacinStudiranja \_nacin;  GodinaStudija \_godina;  Odgovori \_odgovori[brojPitanja];  };  void PrikaziPitanje(int rbPitanja) {  if (rbPitanja == 1)  cout << crt << "Enumeracije i unije su sinonimi ?" << crt;  else if (rbPitanja == 2)  cout << crt << "Moguce je kreirati anonimnu uniju ?" << crt;  else if (rbPitanja == 3)  cout << crt << "Svi clanovi unije koriste istu memorijsku lokaciju?" << crt;  else if (rbPitanja == 4)  cout << crt << "Svi clanovi strukture koriste istu memorijsku lokaciju ? " << crt;  else  cout << crt << "Unija moze biti deklarisana unutar strukture ?" << crt;  }  Odgovori GetTacanOdgovor(int rbPitanja) {  if (rbPitanja == 1)  return NETACNO;  if (rbPitanja == 2)  return TACNO;  if (rbPitanja == 3)  return TACNO;  if (rbPitanja == 4)  return NETACNO;  return TACNO;  }  int GetUspjehStudenta(Student & s) {  int brojacTacnih = 0;  // usporediti odgovore studenta iz niza "s.odgovori[]" sa tačnim odgovorima preko funkcije "GetTacanOdgovor"  for (int i = 0; i < brojPitanja; i++) {  if (s.\_odgovori[i] == GetTacanOdgovor(i+1))  {  brojacTacnih++;  }  }  return brojacTacnih;  }  void PokreniTest(Student & student) {  // prikazati sva pitanja preko funkcije "PrikaziPitanje"  // sa konzole učitati odgovore i evidentirati ih u niz "student.\_odgovori[]"  for (size\_t i = 0; i < brojPitanja; i++)  {  PrikaziPitanje(i+1);  cout << "Unesite odgovor: 1-tacno,2-netacno. " << endl;  int x;  cin >> x;  x--;  student.\_odgovori[i] =(Odgovori)x ;  }  student.\_radioTest = true;  }  void UnosPodataka(Student niz[], int max)  {  int unos;  for (int i = 0; i<max; i++)  {  cout << crt << "\t::STUDENT " << i + 1 << "::" << crt;  cout << "Unesite ime i prezime: ";  cin.getline(niz[i].\_imePrezime, 30);  cout << "Unesite broj indeksa: ";  cin >> niz[i].\_brojIndeksa;  cout << "Nacin studiranja (1.Redovan, 2.DL): ";  //... cin + enum?  int unos1;  cin >> unos1;  NacinStudiranja n = (NacinStudiranja)unos1;  niz[i].\_nacin = n;  cout << "Nacin godinu studija (1,2,3,4): ";  //... cin + enum?  int unos2;  cin >> unos2;  GodinaStudija g = (GodinaStudija)unos2;  niz[i].\_godina = g;  cin.ignore();  niz[i].\_radioTest = false;//tek smo kreirali studenta  //tako da je sigurno da test nije radjen  }  }  int PrikaziMeni() {  int izbor;  do  {  cout << crt << "\t::MENI::" << crt;  cout << "1. Pokreni test" << endl;  cout << "2. Prikazi rezultate" << endl;  cout << "3. Kraj rada" << crt;  cout << "Unesite izbor: ";  cin >> izbor;  } while (izbor<1 || izbor>3);  return izbor;  }  int PrikaziStudente(Student niz[], int max)  {  int rb;  do  {  for (int i = 0; i<max; i++)  cout << i + 1 << ". " << niz[i].\_imePrezime << endl;  cout << crt << "Unesite redni broj studenta: ";  cin >> rb;  } while (rb<1 || rb>max);  return rb - 1;  }  void Pretraga(Student niz[], int max)  {  //definisati funkciju(e) koja(e) ce omoguciti pretragu po:  for (int i = 0; i < max; i++)  {  Student & s = niz[i];  int gs;  cout << "Unesite godinu studija (1,2,3,4)";  cin >> gs;  GodinaStudija g = (GodinaStudija)gs;  cout << "Nacin studiranja (1-Redovan, 2-DL)";  int ns;  cin >> ns;  NacinStudiranja n = (NacinStudiranja)ns;  cout << "Unesite minimalni rezultat %";  int rez;  cin >> rez;    int t = GetUspjehStudenta(s);  int p = 100 \* t / brojPitanja;    bool b1 = s.\_godina==g; //godini studija  bool b2 = s.\_nacin==n; //nacinu studiranja  bool b3 = p>=rez; //min rezultatima ostvarenim na testu  if (b1 && b2 && b3)  cout << s.\_imePrezime << endl;  }  //1. godini studija  //2. nacinu studiranja  //3. rezultatima ostvarenim na testu  //Nove funkcionalnosti dodati u funkciju PrikaziMeni  }  void main()  {  int max, izbor;  cout << "Unesite broj studenata: ";  cin >> max;  cin.ignore();  Student \* niz = new Student[max];  UnosPodataka(niz, max);  system("cls");  do  {  izbor = PrikaziMeni();  if (izbor == 3)  break;  int s = PrikaziStudente(niz, max);  if (izbor == 1)  {  //pokreni test za student s  Student& st = niz[s];  PokreniTest(st);  //niz[s] = st;  }  if (izbor == 2)  {  //prikaži bodove za student s  Student& st = niz[s];  int t = GetUspjehStudenta(st);  int p = 100\* t / brojPitanja;  cout << "Uspjeh je " << p << endl;  }  system("pause>0");  system("cls");  } while (1);  delete[] niz;  niz = NULL;  system("pause");  } |

# Zadatak 3

*U prethodni zadatak implementirati funkciju*

1. *administratoru omoguciti da unosi pitanja te definise tačne odgovore*
2. *administratoru dati prava da studentu ponovo omogući polaganje testa*
3. *uvesti username/password na osnovu koga će se studenti logirati*
4. *odvojiti administratorske funkcije (meni) od korisničkih*
5. *dodati enumeraciju za korisničku ulogu*

|  |
| --- |
| #include <iostream>  #include <fstream>  using namespace std;  char \* crt = "\n========================================\n";  enum NacinStudiranja { REDOVAN, DL };  enum GodinaStudija { PRVA = 1, DRUGA, TRECA, CETVRTA };  enum Odgovori { TACNO, NETACNO };  const int maxPitanja = 100;  int brojPitanja = 0;  struct Student  {  int \_brojIndeksa;  char \_imePrezime[30];  bool \_radioTest;  NacinStudiranja \_nacin;  GodinaStudija \_godina;  Odgovori \_odgovori[brojPitanja];  };  struct Pitanje  {  char tekstPitanja[100];  Odgovori odgovorNaPitanje;  };  Pitanje pitanjaDB[maxPitanja];  void Dodaj(char \*tekstPitanja, Odgovori tacanOdgovor)  {  //  }  void dodajDefaultPitanja()  {  Dodaj("Enumeracije i unije su sinonimi ?", NETACNO);  Dodaj("Moguce je kreirati anonimnu uniju ?", TACNO);  Dodaj("Svi clanovi unije koriste istu memorijsku lokaciju?", TACNO);  Dodaj("Svi clanovi strukture koriste istu memorijsku lokaciju ? ", NETACNO);  Dodaj("Unija moze biti deklarisana unutar strukture ?", TACNO);  }  void PrikaziPitanje(int rbPitanja)  {  }  Odgovori GetTacanOdgovor(int rbPitanja)  {  }  int GetUspjehStudenta(Student & s) {  int brojacTacnih = 0;  // usporediti odgovore studenta iz niza "s.odgovori[]" sa tačnim odgovorima preko funkcije "GetTacanOdgovor"  for (int i = 0; i < brojPitanja; i++) {  if (s.\_odgovori[i] == GetTacanOdgovor(i+1))  {  brojacTacnih++;  }  }  return brojacTacnih;  }  void PokreniTest(Student & student) {  // prikazati sva pitanja preko funkcije "PrikaziPitanje"  // sa konzole učitati odgovore i evidentirati ih u niz "student.\_odgovori[]"  for (size\_t i = 0; i < brojPitanja; i++)  {  PrikaziPitanje(i+1);  cout << "Unesite odgovor: 1-tacno,2-netacno. " << endl;  int x;  cin >> x;  x--;  student.\_odgovori[i] =(Odgovori)x ;  }  student.\_radioTest = true;  }  void UnosPodataka(Student niz[], int max)  {  int unos;  for (int i = 0; i<max; i++)  {  cout << crt << "\t::STUDENT " << i + 1 << "::" << crt;  cout << "Unesite ime i prezime: ";  cin.getline(niz[i].\_imePrezime, 30);  cout << "Unesite broj indeksa: ";  cin >> niz[i].\_brojIndeksa;  cout << "Nacin studiranja (1.Redovan, 2.DL): ";  //... cin + enum?  int unos1;  cin >> unos1;  NacinStudiranja n = (NacinStudiranja)unos1;  niz[i].\_nacin = n;  cout << "Nacin godinu studija (1,2,3,4): ";  //... cin + enum?  int unos2;  cin >> unos2;  GodinaStudija g = (GodinaStudija)unos2;  niz[i].\_godina = g;  cin.ignore();  niz[i].\_radioTest = false;//tek smo kreirali studenta  //tako da je sigurno da test nije radjen  }  }  int PrikaziMeni() {  int izbor;  do  {  cout << crt << "\t::MENI::" << crt;  cout << "1. Pokreni test" << endl;  cout << "2. Prikazi rezultate" << endl;  cout << "3. Kraj rada" << crt;  cout << "Unesite izbor: ";  cin >> izbor;  } while (izbor<1 || izbor>3);  return izbor;  }  int PrikaziStudente(Student niz[], int max)  {  int rb;  do  {  for (int i = 0; i<max; i++)  cout << i + 1 << ". " << niz[i].\_imePrezime << endl;  cout << crt << "Unesite redni broj studenta: ";  cin >> rb;  } while (rb<1 || rb>max);  return rb - 1;  }  void Pretraga(Student niz[], int max)  {  //definisati funkciju(e) koja(e) ce omoguciti pretragu po:  for (int i = 0; i < max; i++)  {  Student & s = niz[i];  int gs;  cout << "Unesite godinu studija (1,2,3,4)";  cin >> gs;  GodinaStudija g = (GodinaStudija)gs;  cout << "Nacin studiranja (1-Redovan, 2-DL)";  int ns;  cin >> ns;  NacinStudiranja n = (NacinStudiranja)ns;  cout << "Unesite minimalni rezultat %";  int rez;  cin >> rez;    int t = GetUspjehStudenta(s);  int p = 100 \* t / brojPitanja;    bool b1 = s.\_godina==g; //godini studija  bool b2 = s.\_nacin==n; //nacinu studiranja  bool b3 = p>=rez; //min rezultatima ostvarenim na testu  if (b1 && b2 && b3)  cout << s.\_imePrezime << endl;  }  //1. godini studija  //2. nacinu studiranja  //3. rezultatima ostvarenim na testu  //Nove funkcionalnosti dodati u funkciju PrikaziMeni  }  void main()  {  int max, izbor;  cout << "Unesite broj studenata: ";  cin >> max;  cin.ignore();  Student \* niz = new Student[max];  UnosPodataka(niz, max);  system("cls");  do  {  izbor = PrikaziMeni();  if (izbor == 3)  break;  int s = PrikaziStudente(niz, max);  if (izbor == 1)  {  //pokreni test za student s  Student& st = niz[s];  PokreniTest(st);  //niz[s] = st;  }  if (izbor == 2)  {  //prikaži bodove za student s  Student& st = niz[s];  int t = GetUspjehStudenta(st);  int p = 100\* t / brojPitanja;  cout << "Uspjeh je " << p << endl;  }  system("pause>0");  system("cls");  } while (1);  delete[] niz;  niz = NULL;  system("pause");  } |

|  |
| --- |
| **Rješenje 3.1** |
| #include <iostream>  #include <fstream>  using namespace std;  char \* crt = "\n========================================\n";  enum NacinStudiranja { REDOVAN, DL };  enum GodinaStudija { PRVA = 1, DRUGA, TRECA, CETVRTA };  enum Odgovori { TACNO, NETACNO };  const int maxPitanja = 100;  int brojPitanja = 0;  struct Student  {  int \_brojIndeksa;  char \_imePrezime[30];  bool \_radioTest;  NacinStudiranja \_nacin;  GodinaStudija \_godina;  Odgovori \_odgovori[maxPitanja];  };  struct Pitanje  {  char tekstPitanja[100];  Odgovori odgovorNaPitanje;  };  Pitanje pitanjaDB[maxPitanja];  void Dodaj(char \*tekstPitanja, Odgovori tacanOdgovor)  {  //  Pitanje nekoPitanje;  strcpy\_s(nekoPitanje.tekstPitanja, tekstPitanja);  nekoPitanje.odgovorNaPitanje = tacanOdgovor;  pitanjaDB[brojPitanja] = nekoPitanje;  brojPitanja++;  }  void dodajDefaultPitanja()  {  Dodaj("Enumeracije i unije su sinonimi ?", NETACNO);  Dodaj("Moguce je kreirati anonimnu uniju ?", TACNO);  Dodaj("Svi clanovi unije koriste istu memorijsku lokaciju?", TACNO);  Dodaj("Svi clanovi strukture koriste istu memorijsku lokaciju ? ", NETACNO);  Dodaj("Unija moze biti deklarisana unutar strukture ?", TACNO);  }  void PrikaziPitanje(int rbPitanja)  {  cout << pitanjaDB[rbPitanja].tekstPitanja << endl;  }  Odgovori GetTacanOdgovor(int rbPitanja)  {  return pitanjaDB[rbPitanja].odgovorNaPitanje;  }  int GetUspjehStudenta(Student & s) {  int brojacTacnih = 0;  // usporediti odgovore studenta iz niza "s.odgovori[]" sa tačnim odgovorima preko funkcije "GetTacanOdgovor"  for (int i = 0; i < brojPitanja; i++) {  if (s.\_odgovori[i] == GetTacanOdgovor(i+1))  {  brojacTacnih++;  }  }  return brojacTacnih;  }  void PokreniTest(Student & student) {  // prikazati sva pitanja preko funkcije "PrikaziPitanje"  // sa konzole učitati odgovore i evidentirati ih u niz "student.\_odgovori[]"  for (size\_t i = 0; i < brojPitanja; i++)  {  PrikaziPitanje(i);  cout << "Unesite odgovor: 1-tacno,2-netacno. " << endl;  int x;  cin >> x;  x--;  student.\_odgovori[i] =(Odgovori)x ;  }  student.\_radioTest = true;  }  void UnosPodataka(Student niz[], int max)  {  int unos;  for (int i = 0; i<max; i++)  {  cout << crt << "\t::STUDENT " << i + 1 << "::" << crt;  cout << "Unesite ime i prezime: ";  cin.getline(niz[i].\_imePrezime, 30);  cout << "Unesite broj indeksa: ";  cin >> niz[i].\_brojIndeksa;  cout << "Nacin studiranja (1.Redovan, 2.DL): ";  //... cin + enum?  int unos1;  cin >> unos1;  NacinStudiranja n = (NacinStudiranja)unos1;  niz[i].\_nacin = n;  cout << "Nacin godinu studija (1,2,3,4): ";  //... cin + enum?  int unos2;  cin >> unos2;  GodinaStudija g = (GodinaStudija)unos2;  niz[i].\_godina = g;  cin.ignore();  niz[i].\_radioTest = false;//tek smo kreirali studenta  //tako da je sigurno da test nije radjen  }  }  int PrikaziMeni() {  int izbor;  do  {  cout << crt << "\t::MENI::" << crt;  cout << "1. Pokreni test" << endl;  cout << "2. Prikazi rezultate" << endl;  cout << "3. Dodaj novo pitanje" << crt;  cout << "4. Kraj rada" << crt;  cout << "Unesite izbor: ";  cin >> izbor;  } while (izbor<1 || izbor>4);  return izbor;  }  int PrikaziStudente(Student niz[], int max)  {  int rb;  do  {  for (int i = 0; i<max; i++)  cout << i + 1 << ". " << niz[i].\_imePrezime << endl;  cout << crt << "Unesite redni broj studenta: ";  cin >> rb;  } while (rb<1 || rb>max);  return rb - 1;  }  void Pretraga(Student niz[], int max)  {  //definisati funkciju(e) koja(e) ce omoguciti pretragu po:  for (int i = 0; i < max; i++)  {  Student & s = niz[i];  int gs;  cout << "Unesite godinu studija (1,2,3,4)";  cin >> gs;  GodinaStudija g = (GodinaStudija)gs;  cout << "Nacin studiranja (1-Redovan, 2-DL)";  int ns;  cin >> ns;  NacinStudiranja n = (NacinStudiranja)ns;  cout << "Unesite minimalni rezultat %";  int rez;  cin >> rez;    int t = GetUspjehStudenta(s);  int p = 100 \* t / brojPitanja;    bool b1 = s.\_godina==g; //godini studija  bool b2 = s.\_nacin==n; //nacinu studiranja  bool b3 = p>=rez; //min rezultatima ostvarenim na testu  if (b1 && b2 && b3)  cout << s.\_imePrezime << endl;  }  //1. godini studija  //2. nacinu studiranja  //3. rezultatima ostvarenim na testu  //Nove funkcionalnosti dodati u funkciju PrikaziMeni  }  void main()  {  int max, izbor;  cout << "Unesite broj studenata: ";  cin >> max;  cin.ignore();  Student \* niz = new Student[max];  UnosPodataka(niz, max);  system("cls");  dodajDefaultPitanja();  do  {  izbor = PrikaziMeni();  if (izbor == 4)  break;    if (izbor == 1)  {  int s = PrikaziStudente(niz, max);  //pokreni test za student s  Student& st = niz[s];  PokreniTest(st);  //niz[s] = st;  }  if (izbor == 2)  {  int s = PrikaziStudente(niz, max);  //prikaži bodove za student s  Student& st = niz[s];  int t = GetUspjehStudenta(st);  int p = 100\* t / brojPitanja;  cout << "Uspjeh je " << p << endl;  }  if (izbor == 3)  {  cout << "tekst pitanja" << endl;  char p[100];  cin.ignore();  cin.getline(p, 100);  int x;  cout << "Tacno 1, Netacno 2" << endl;  cin >> x;  Odgovori g = (Odgovori)x;  Dodaj(p, g);  }  system("pause>0");  system("cls");  } while (1);  delete[] niz;  niz = NULL;  system("pause");  } |

# Zadatak 4 - *Zadatak sa kolekcijama i lambda funkcijama.*

*U nastavku je dato rješenje zadnjeg zadatka iz prethodne sedmice:*

|  |
| --- |
| #include <iostream>  #include <functional>  #include <algorithm>  using namespace std;  const int max\_size = 30;  struct Kolekcija  {  int podaci[max\_size];  int brojac = 0;  };  void kDodaj(Kolekcija& k, int newV)  {  //dodaj newV u podaci na poziciju brojac  //uvecati brojac  k.podaci[k.brojac++] = newV;  }  void kGenerisiPodatke(Kolekcija& k)  {  for (int i = 0; i < max\_size / 2; i++)  {  int newV = rand() - rand();  //newV dodati u kolekciju k  kDodaj(k, int(newV));  }  }  void kPrinter(Kolekcija& k, char\* opis)  {  cout << opis << endl;  for (int i = 0; i < k.brojac; i++)  {  int& p = k.podaci[i];  cout << i << ": " << p << endl;  }  }  Kolekcija kFilter(Kolekcija& k, function<bool(int&)> f)  {  Kolekcija newK;  //iz kolekcije k u kolekciju newK kopirati elemente koji zadovoljavaju uslov f  for (size\_t i = 0; i < k.brojac; i++)  {  int &x = k.podaci[i];  if (f(x)) {  kDodaj(newK, x);  }  }  return newK;  }  void kForEach(Kolekcija& k, function<void(int&)> f)  {  //za svaki element iz kolekcije k izvrsi zadatak f  for (int i = 0; i < k.brojac; i++)  {  int &x = k.podaci[i];  f(x);  }  }  void main()  {  //a. Instanciraj Kolekciju k1 i generiši podatke  Kolekcija k1;  kGenerisiPodatke(k1);  //b. Iz kolekcije k1 filtiraj parne brojeve u kolekciju k2  Kolekcija k2 = kFilter(k1, [](int &b) { return b % 2 == 0; });  //c. Iz kolekcije k2 filtiraj pozitivne brojeve u kolekciju k3  Kolekcija k3 = kFilter(k2, [](int &b) { return b>0; });  //d. Ispisati na ekran sve tri kolekcije  kPrinter(k1, "kolekcija 1 ");  kPrinter(k2, "kolekcija 2 ");  kPrinter(k3, "kolekcija 3 ");  //f. koristeci kForEach uvecati sve elemente kolekcije k3  kForEach(k3, [](int &b) {b++; });  //g. koristeci kForEach kopirati sve elemente kolekcije k2 u novu kolekciju  Kolekcija k4;  kForEach(k2, [&k4](int &v) {kDodaj(k4, v); });  } |

1. *Dodati* ***typedef Tip*** *za integer, te promjeniti cijeli code tako da se koristi* ***Tip***
2. *Prepravii kolekciju tako da se koristi* ***shared\_ptr*** *kao elemenat niza kolekcija*
3. *nastavak*
   1. *koristeci* ***kForEach*** *izračunati sumu elemenata k4*
   2. *koristeci* ***kForEach*** *zamjeniti negativne elemente sa 0*
   3. *koristeci* ***kForEach*** *sve elemente k4 zamjeniti sa vrijednostima njihovog korijena*

|  |
| --- |
| **Rješenje – varijanta 1: typedef shared\_ptr** |
| #include <iostream>  #include <functional>  #include <algorithm>  #include <memory>  using namespace std;  typedef shared\_ptr<int> Tip;  const int max\_size = 30;  struct Kolekcija  {  Tip podaci[max\_size];  int brojac = 0;  };  void kDodaj(Kolekcija& k, Tip newV)  {  //dodaj newV u podaci na poziciju brojac  //uvecati brojac  k.podaci[k.brojac++] = newV;  }  void kGenerisiPodatke(Kolekcija& k)  {  for (int i = 0; i < max\_size / 2; i++)  {  int newV = rand() - rand();  //newV dodati u kolekciju k  kDodaj(k, make\_shared<int>(newV));  }  }  void kPrinter(Kolekcija& k, char\* opis)  {  cout << opis << endl;  for (int i = 0; i < k.brojac; i++)  {  Tip& p = k.podaci[i];  cout << i << ": " << \*p << endl;  }  }  Kolekcija kFilter(Kolekcija& k, function<bool(Tip&)> f)  {  Kolekcija newK;  //iz kolekcije k u kolekciju newK kopirati elemente koji zadovoljavaju uslov f  for (size\_t i = 0; i < k.brojac; i++)  {  Tip &x = k.podaci[i];  if (f(x)) {  kDodaj(newK, x);  }  }  return newK;  }  void kForEach(Kolekcija& k, function<void(Tip&)> f)  {  //za svaki element iz kolekcije k izvrsi zadatak f  for (int i = 0; i < k.brojac; i++)  {  Tip &x = k.podaci[i];  f(x);  }  }  void main()  {  //a. Instanciraj Kolekciju k1 i generiši podatke  Kolekcija k1;  kGenerisiPodatke(k1);  //b. Iz kolekcije k1 filtiraj parne brojeve u kolekciju k2  Kolekcija k2 = kFilter(k1, [](Tip &b) { return \*b % 2 == 0; });  //c. Iz kolekcije k2 filtiraj pozitivne brojeve u kolekciju k3  Kolekcija k3 = kFilter(k2, [](Tip &b) { return \*b>0; });  //d. Ispisati na ekran sve tri kolekcije  kPrinter(k1, "kolekcija 1 ");  kPrinter(k2, "kolekcija 2 ");  kPrinter(k3, "kolekcija 3 ");  //f. koristeci kForEach uvecati sve elemente kolekcije k3  kForEach(k3, [](Tip &b) {(\*b)++; });  //g. koristeci kForEach kopirati sve elemente kolekcije k2 u novu kolekciju  Kolekcija k4;  kForEach(k2, [&k4](Tip &v) {kDodaj(k4, v); });  //h. koristeci kForEach izračunati sumu elemenata k4  int s = 0;  kForEach(k4, [&s](Tip &v) {s += \*v; });  //i. koristeci kForEach zamjeniti negativne elemente sa 0  kForEach(k4, [](Tip &v) { if (v<0) v = 0; });  //j. koristeci kForEach sve elemente k4 zamjeniti sa vrijednostima njihovog korijena  kForEach(k4, [](Tip &v) { \*v = sqrt(\*v); });  } |

|  |
| --- |
| **Rješenje – varijanta 2: niz tipa shared\_ptr** |
| #include <iostream>  #include <functional>  #include <algorithm>  #include <memory>  using namespace std;  typedef int Tip;  const int max\_size = 30;  struct Kolekcija  {  shared\_ptr<Tip> podaci[max\_size];  int brojac = 0;  };  void kDodaj(Kolekcija& k, Tip newV)  {  //dodaj newV u podaci na poziciju brojac  //uvecati brojac  k.podaci[k.brojac++] = make\_shared<Tip>(newV);  }  void kGenerisiPodatke(Kolekcija& k)  {  for (int i = 0; i < max\_size / 2; i++)  {  int newV = rand() - rand();  //newV dodati u kolekciju k  kDodaj(k, newV);  }  }  void kPrinter(Kolekcija& k, char\* opis)  {  cout << opis << endl;  for (int i = 0; i < k.brojac; i++)  {  shared\_ptr<Tip>& p = k.podaci[i];  cout << i << ": " << \*p << endl;  }  }  Kolekcija kFilter(Kolekcija& k, function<bool(Tip&)> f)  {  Kolekcija newK;  //iz kolekcije k u kolekciju newK kopirati elemente koji zadovoljavaju uslov f  for (size\_t i = 0; i < k.brojac; i++)  {  shared\_ptr<Tip> &x = k.podaci[i];  if (f(\*x)) {  kDodaj(newK, \*x);  }  }  return newK;  }  void kForEach(Kolekcija& k, function<void(Tip&)> f)  {  //za svaki element iz kolekcije k izvrsi zadatak f  for (int i = 0; i < k.brojac; i++)  {  shared\_ptr<Tip> &x = k.podaci[i];  f(\*x);  }  }  void main()  {  //a. Instanciraj Kolekciju k1 i generiši podatke  Kolekcija k1;  kGenerisiPodatke(k1);  //b. Iz kolekcije k1 filtiraj parne brojeve u kolekciju k2  Kolekcija k2 = kFilter(k1, [](Tip &b) { return b % 2 == 0; });  //c. Iz kolekcije k2 filtiraj pozitivne brojeve u kolekciju k3  Kolekcija k3 = kFilter(k2, [](Tip &b) { return b>0; });  //d. Ispisati na ekran sve tri kolekcije  kPrinter(k1, "kolekcija 1 ");  kPrinter(k2, "kolekcija 2 ");  kPrinter(k3, "kolekcija 3 ");  //f. koristeci kForEach uvecati sve elemente kolekcije k3  kForEach(k3, [](Tip &b) {b++; });  //g. koristeci kForEach kopirati sve elemente kolekcije k2 u novu kolekciju  Kolekcija k4;  kForEach(k2, [&k4](Tip &v) {kDodaj(k4, v); });  //h. koristeci kForEach izračunati sumu elemenata k4  int s = 0;  kForEach(k4, [&s](Tip &v) {s += v; });  //i. koristeci kForEach zamjeniti negativne elemente sa 0  kForEach(k4, [](Tip &v) { if (v<0) v = 0; });  //j. koristeci kForEach sve elemente k4 zamjeniti sa vrijednostima njihovog korijena  kForEach(k4, [](Tip &v) { v = sqrt(v); });  } |

Autor:

mr. Adil Joldić

[adil@edu.fit.ba](mailto:adil@edu.fit.ba)