|  |  |  |
| --- | --- | --- |
| **not safe** | **safe** | |
| **char \* D = new char[x];**  **char D[10];** | **char D[10];** | **char \* D = new char[**size\_d**];** |
| strcpy(D, S) | strcpy\_s(D, S) | strcpy\_s(D, size\_d, S) |
| str**n**cpy(D, S, size\_s) | str**n**cpy\_s(D, S, size\_s) | str**n**cpy\_s(D, size\_d, S, size\_s) |

Prepravite naredne kodove tako da koristite safe funkcije (strcpy\_s, strncpy\_s, strcat\_s).

# Zadatak 1.

Šta je rezultat slijedeća tri primjera.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Primjer* ***a****:* niz *destination* je veći od niza *source*   |  |  | | --- | --- | | s je dinamički niz  d je statički niz | d je dinamički niz  s je statički niz | | char\* s = new char[7];  cin >> s;  char d[13];  strcpy(d, s);  cout << strlen(s) << endl;  cout << strlen(d) << endl; | char s[7];  cin >> s;  char\* d = new char[13];  strcpy(d, s);  cout << strlen(s) << endl;  cout << strlen(d) << endl; | |
| *Primjer* ***b****:* korištenje funkcije strncpy   |  |  | | --- | --- | | s je dinamički niz  d je statički niz | d je dinamički niz  s je statički niz | | char\* s = new char[7];  cin >> s;  char d[13];  strncpy(d, s, 3);  cout << strlen(s) << endl;  cout << strlen(d) << endl; | char s[7];  cin >> s;  char\* d = new char[13];  strncpy(d, s, 3);  cout << strlen(s) << endl;  cout << strlen(d) << endl; |     Ovdje je potrebno dodati karakter '\0' na kraj stringa. Slijedi ispravak.   |  |  | | --- | --- | | char\* s = new char[7];  cin >> s;  char d[13];  strncpy(d, s, 3);  **d[3] = '\0';**  cout << strlen(s) << endl;  cout << strlen(d) << endl; | char s[7];  cin >> s;  char\* d = new char[13];  strncpy(d, s, 3);  **d[3] = '\0';**  cout << strlen(s) << endl;  cout << strlen(d) << endl; | |

|  |  |  |
| --- | --- | --- |
| *Primjer* ***c****:* niz *destination* je manji od niza *source*, ali nije manji od *stringa* iz *source*  *Da li će doći do greške?*   |  |  | | --- | --- | | char\* s = new char[13];  cin >> s;  char d[9];  strcpy(d, s);  cout << strlen(s) << endl;  cout << strlen(d) << endl; | char s[13];  cin >> s;  char\* d = new char[9];  strcpy(d, s);  cout << strlen(s) << endl;  cout << strlen(d) << endl; | |

# Zadatak 2

*Pitanje:* Šta će ispisati sljedeći kôd?

|  |
| --- |
| char p [20];  cout << strlen(p) << endl; |

# Zadatak 3

* Definišite pokazivač A na niz karaktera u dinamičkoj memoriji u kojeg čete smjestiti ime i prezime.
* Definišite statički niz B dužine 20 u koji ćete unjeti mjesto rođenja
* A i B kopirajte u novi dinamički niz karaktera C.
* Reciklirajte oba dinamička niza.

|  |  |
| --- | --- |
| 1:  2:  3:  4:  5:  6:  7:  8:  9:  10:  11:  12:  13:  14:  15:  16:  17:  18:  19:  20:  21:  22:  23:  24: | #include <iostream>  using namespace std;  void main()  {  char\* A = new char[30];  char B[20];  cout << "Unesi ime i prezime: \n";  cin.getline(A, 29); // maksimalna duzina je 29, jer je potrebno jedno mjesto za '\0'  cout << "Unesi mjesto rođenja: \n";  cin.getline(B, 19);  // duzina A + razmak + duzina B + '\n'  int PotrebnaDuzina = strlen(A) + 1 + strlen(B) + 1;  char\* C = new char [PotrebnaDuzina];  strcpy(C, A); // A kopira u C  strcat(C, " "); // u C dodaje "razmak", a mogao je biti bilo koji tekst  strcat(C, B); // u C dodaje B    cout << C << endl;  delete [] A;  delete [] C;  } |

# Zadatak 4

Analizirajte sljedeći program.

|  |  |
| --- | --- |
| 1:  2:  3:  4:  5:  6:  7:  8:  9:  10:  11:  12:  13:  14:  15:  16: | void main()  {  char\* str1 = "ovo je prvi string u statickoj memoriji";  char str2[] = "ovo je drugi string u statickoj memoriji";    char\* p1;  char\* p2;  p1 = str1; //p1 pokazuje na string str1  p2 = str2; //p2 pokazuje na string str2  cout << str1 << endl;  cout << str2 << endl;  cout << p1 << endl;  cout << p2 << endl;  } |

Pitanje: Da li će se u linijama 11, 12, 13 i 14 ispisati adresa prvog elementa niza? Obrazložite.

# Zadatak 5

Dovršite naredni program:

1. ispisati dati string od trećeg karaktera
2. ispisati prvi karakter stringa
3. ispisati zadnji karakter stringa
4. ispisati zadnjih 10 karaktera stringa

|  |  |
| --- | --- |
| 1:  2:  3: | void main()  {  char\* str1 = "ovo je prvi string u statickoj memoriji";  ... |

|  |  |
| --- | --- |
| **Rješenje** | |
| 1:  2:  3:  4:  5:  6:  7:  8:  9:  10:  11:  12:  13:  14: | void main()  {  char\* str1 = "ovo je string u statickoj memoriji";    // a)  cout << (str1 + 3) << endl;  // b)  cout << \*str1 << endl; //isto kao str1[0]  // c)  int d = strlen(str1);  cout << str1[d-1] << endl; // isto kao \*(str1 + d - 1)  // d)  cout << (str1 + d - 10) << endl;  } |

# Zadatak 6

Deklarišite dva stringa (str1 i str2). Zatim ih iskopirajte (spojite) u treći string str3 i ispišite ga na ekran.

1. koristite funkciju strcat
2. koristite funkciju strcpy (bez korištenja funkcije strcat )



|  |  |
| --- | --- |
| **Rješenje - a** | |
| 1:  2:  3:  4:  5:  6:  7:  8:  9:  10:  11:  12:  13:  14:  15:  16:  17: | void main()  {  char\* str1 = "Ovo je prvi string. ";  char\* str2 = "Ovo je drugi string.";  char\* str3;  int d1 = strlen(str1);  int d2 = strlen(str2);  str3 = new char [d1 + d2 + 1];  strcpy(str3, str1);  strcat(str3, str2);  cout << str3 << endl;  delete [] str3;  } |

|  |  |
| --- | --- |
| **Rješenje - b** | |
| 1:  2:  3:  4:  5:  6:  7:  8:  9:  10:  11:  12:  13:  14:  15:  16:  17: | void main()  {  char\* str1 = "prvi string.";  char\* str2 = "drugi string.";  char\* str3;  int d1 = strlen(str1); //12  int d2 = strlen(str2); //13  str3 = new char [d1 + d2 + 1];  strcpy(str3 + 0, str1);  strcpy(str3 + d1, str2); // strcat(str3, str2);  cout << str3 << endl;  delete [] str3;  } |

# Zadatak 8

1. Implementirajte funkciju dodjeli\_str koja će alocirati niz u dinamičkoj memoriji i kopirati ulazni string u taj niz, npr.

|  |  |
| --- | --- |
| 1:  2:  3:  4:  5:  6:  7:  8:  9: | void main()  {  char\* str1 = dodjeli\_str("Ovo je neki string");  //funkcija vrši alokaciju i kopiranje  cout << str1 << endl;  delete [] str1; //dealokacija  } |

|  |  |
| --- | --- |
| Rješenje: | |
| 1:  2:  3:  4:  5:  6:  7:  8:  9:  10:  11:  12:  13:  14:  15:  16:  17:  18: | #include <iostream>  using namespace std;  char\* dodjeli\_str(char\* s)  {  char\* x = new char[strlen(s) + 1];  strcpy(x, s);  return x;  }  void main()  {  char\* str1;  str1 = dodjeli\_str("Ovo je neki string");  cout << str1 << endl;  delete [] str1;  } |

1. Primjenite funkciju dodjeli u sljedećem programu.

|  |  |
| --- | --- |
| 1:  2:  3:  4:  5:  6:  7:  8:  9:  10:  11:  12:  13:  14:  15: | void main()  {  char\* str1;  **str1 = new char[strlen("Ovo je neki string") + 1];**  **strcpy(str1, "Ovo je neki string");**  cout << str1 << endl;  **char\* str2 = new char[strlen(str1) + 1];**  **strcpy(str2, str1);**  cout << str2 << endl;  delete [] str1;  delete [] str2;  } |

# Zadatak 9

Implementirajte funkciju dodaj\_str koja će prvom stringu dodati drugi string.

|  |  |
| --- | --- |
| 1:  2:  3:  4:  5:  6:  7:  8: | void main()  {  char\* str1 = dodjeli\_str("Prvi niz. ");  dodaj\_str(str1, "Ovo je drugi niz");  cout << str1 << endl; //ispis: "Prvi niz. Ovo je drugi niz."  delete [] str1;  } |

|  |  |
| --- | --- |
| 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: | #include <iostream>  using namespace std;  char\* dodjeli\_str(char\* s)  {  char\* x = new char[strlen(s) + 1];  strcpy(x, s);  return x;  }  void dodaj\_str(char\* &a, char\* b)  {  int d1 = strlen(a);  int d2 = strlen(b);  char\* x = new char[d1 + d2 + 1]; //alociramo novi niz tako da u njega moze stati niz i niz b  strcpy(x, a); // u novi niz kopiramo a  strcat(x, b); // u novi niz dodajemo b  delete [] a; //alociramo stari niz a  a = x; //neka a pokazuje na novi niz  }  void main()  {  char\* str1 = dodjeli\_str("Prvi niz. ");  dodaj\_str(str1, "Ovo je drugi niz");  cout << str1 << endl;  delete [] str1;  } |

# Zadatak 10:

Definišite funkciju "void dodjeli\_str(char\* &d, char\* b)" koja ima istu funkcionalnost kao "char\* dodjeli\_str(char\* s)" s tim da izlaz bude pomoću reference a ne pomoću naredbe return.

Nemojte brisati prvu definiciju funkcije, jer je moguće imati više funkcija sa istim imenom a koje se razliku po broju ili tipu parametara.

# Zadatak 11:

1. Definište funkciju spoji\_str koja će kreirati izlazni string sastavljen od dva ulazna stringa. Izlazna vrijednost neka bude pomoću naredbe return.

|  |  |
| --- | --- |
| Rješenje | |
| 1:  2:  3:  4:  5:  6:  7: | char\* spoji\_str(char\* a, char\* b)  {  int d1 = strlen(a);  int d2 = strlen(b);  char\* x = new char[d1 + d2 + 1];  strncpy(x, a, d1 + 1);  strncat(x, b, d2 + 1);  return x;  } |

1. Ponovo definište funkciju spoji\_str koja će kreirati izlazni string sastavljen od tri ulazna stringa. Izlazna vrijednost neka bude pomoću naredbe return.

|  |  |
| --- | --- |
| Rješenje | |
| 1:  2:  3:  4:  5:  6:  7:  8: | char\* spoji\_str(char\* a, char\* b, char\* c)  {  int d1 = strlen(a);  int d2 = strlen(b);  int d3 = strlen(c);  char\* x = new char[d1 + d2 + d3 + 1];  strncpy(x, a, d1 + 1);  strncat(x, b, d2 + 1);  strncat(x, c, d3 + 1);  return x;  } |

Kopirajte vaše četiri funkcije sa njihovim prototipovima u jedan fajl i taj fajl snimite u folder gdje se nalaze i drugi *header*-fajlovi ili negdje na disk tako da možete ubuduće koristiti vaše funkcije kao gotove funkcije, npr.:

|  |  |
| --- | --- |
| 1:  2:  3:  4:  5:  6:  7:  8:  9:  10:  11:  12:  13:  14:  15: | #include <iostream>  using namespace std;  #include "c:\temp\str\_funkcije.h"  void main()  {  char\* p1 = dodjeli\_str("\_Fit for FIT");    char\* p2;  dodjeli\_str(p2, "He says:");  dodaj\_str(p2, p1);  cout << "p2 = " << p2 << endl;    char\* p3 = spoji\_str(p2, " > ", p1);  //...nemojte zaboraviti dealocirati nizove p1, p2 i p3... |

# Zadatak 12 – Tekst Editor

Dovršite program

|  |
| --- |
| #include <iostream>  using namespace std;  char crt[] = "\n--------------------------------------------------\n";  void OslobodiMemoriju(char \* &tekst)  {  //dealocirati tekst  }  void Informacije(char \* tekst)  {  int razmaci = 0, brojevi = 0, velika = 0, mala = 0, interpunkcijski = 0;    //...  cout << crt << "\t\t::INFO::" << crt;  cout << crt << "Tekst: " << tekst << crt;  cout << "Niz ima " << strlen(tekst) << " karaktera." << crt;  cout << "Razmaka: \t\t\t" << razmaci << endl;  cout << "Brojeva: \t\t\t" << brojevi << endl;  cout << "Velikih slova: \t\t\t" << velika << endl;  cout << "Malih slova: \t\t\t" << mala << endl;  cout << "Interpunkcijskih znakova: \t" << interpunkcijski;  cout << crt << "Info: Informacije prikazane...." << crt;  }    void DodajTekst(char \* &tekst)  {  //stari tekst + razmak + novi tekst  }  void Pretraga(char \* tekst)  {  //...  }  void UnosTeksta(char \*& tekst)  {  //...  }  int PrikaziMeni() {  int izbor = 1;  do {  cout << crt << "\t\t::MENI::" << crt;  cout << "1. Unos novog teksta" << endl;  cout << "2. Dodavanje teksta" << endl;  cout << "3. Informacije o tekstu" << endl;  cout << "4. Pretraga" << endl;  cout << "5. Zatvori editor" << endl;  cout << "Unesite vas izbor: ";  cin >> izbor;  cin.ignore();  system("cls");  } while (izbor < 1 || izbor>5);  return izbor;  }  void main() {  int izbor = 0;  char \* tekst = NULL;  do {  cout << crt << "\t\t::TEKST EDITOR::";  izbor = PrikaziMeni();  switch (izbor) {  case 1:  UnosTeksta(tekst); break;  case 2:  DodajTekst(tekst); break;  case 3:  Informacije(tekst); break;  case 4:  Pretraga(tekst); break;  }  system("pause>0");  system("cls");  } while (izbor != 5);    if (tekst != NULL)  OslobodiMemoriju(tekst);  cout << crt;  } |

|  |
| --- |
| #include <iostream>  using namespace std;  char crt[] = "\n--------------------------------------------------\n";  //za vjezbu pokusajte otkriti razlog zasto funkcija prihvata  //adresu pokazivaca!  void OslobodiMemoriju(char \* &tekst) {  delete[] tekst;  tekst = NULL;  cout << crt << "Info: Memorija oslobodjena!" << crt;  }  void Informacije(char \* tekst) {  int razmaci = 0, brojevi = 0, velika = 0, mala = 0, interpunkcijski = 0;  for (int i = 0; i < strlen(tekst); i++)  {  if (isspace(tekst[i]))  razmaci++;  else if (isdigit(tekst[i]))  brojevi++;  else if (isupper(tekst[i]))  velika++;  else if (islower(tekst[i]))  mala++;  else if (ispunct(tekst[i]))  interpunkcijski++;  }  cout << crt << "\t\t::INFO::" << crt;  cout << crt << "Tekst: " << tekst << crt;  cout << "Niz ima " << strlen(tekst) << " karaktera." << crt;  cout << "Razmaka: \t\t\t" << razmaci << endl;  cout << "Brojeva: \t\t\t" << brojevi << endl;  cout << "Velikih slova: \t\t\t" << velika << endl;  cout << "Malih slova: \t\t\t" << mala << endl;  cout << "Interpunkcijskih znakova: \t" << interpunkcijski;  cout << crt << "Info: Informacije prikazane...." << crt;  }  //za vjezbu pokusajte otkriti razlog zasto funkcija prihvata  //adresu pokazivaca!  void DodajTekst(char \* &tekst) {  const int max = 300;  char temp[max];  cout << crt << "\t\t::DODAJ::" << crt;  cout << tekst << " ";//ispisujemo trenutni tekst  cin.getline(temp, max);  //alociramo novu memoriju na osnovu velicine postojeceg i novog teksta  //broj dva smo dodali velicini zbog; razmaka izmedju starog i novog teksta te  //null terminirajuceg karaktera  char \* noviTekst = new char[strlen(tekst)+strlen(temp)+2];  strcpy(noviTekst, tekst);//prvo kopiramo stari tekst u novi niz  strcat(noviTekst," "); //dodajemo razmak  strcat(noviTekst,temp); //nizu dodajemo novi tekst koji je korisnik unio  OslobodiMemoriju(tekst);//brisemo stari tekst  tekst = noviTekst;//pokazivac tekst sada pokazuju na novi niz koji sadrzi  //stari tekst + novi tekst  cout << crt << "Info: Novi tekst dodat...." << crt;  }  void Pretraga(char \* tekst) {  const int max = 20;  char temp[max];  char nastavak;  cout << crt << "\t\t::PRETRAGA::" << crt; cout << "Unesite tekst koji trazite: ";  cin.getline(temp, max);  //provjeravamo da li trazena rijec postoji u tekst  char \* pok = strstr(tekst, temp);  if (pok == NULL)  cout << "Tekst \'" << temp << "\' nije pronadjen!" << crt;  else {  cout << "Tekst \'" << temp << "\' je pronadjen!" << endl;  cout << "Da li zelite prikazati sadrzaj koji se nalazi nakon pronadjenog teksta(D/N):";  cin >> nastavak;  if (toupper(nastavak) == 'D')  cout << crt << "Tekst: " << crt << pok << crt;  }  }  void UnosTeksta(char \*& tekst) {  const int max = 300;  char temp[max];  cout << "Unesite vas tekst (#-kraj): ";  cin.getline(temp, max, '#');  if (tekst != NULL)//ako se ne radi o prvom unosu  OslobodiMemoriju(tekst);//dealociraj memoriju  //alociraj memoriju na osnovu velicine unesenog teksta (temp) tekst = new char[strlen(temp)+1];  strcpy(tekst, temp);  }  int PrikaziMeni() {  int izbor = 1;  do {  cout << crt << "\t\t::MENI::" << crt;  cout << "1. Unos novog teksta" << endl;  cout << "2. Dodavanje teksta" << endl;  cout << "3. Informacije o tekstu" << endl;  cout << "4. Pretraga" << endl;  cout << "5. Zatvori editor" << endl;  cout << "Unesite vas izbor: ";  cin >> izbor;  cin.ignore();  system("cls");  } while (izbor < 1 || izbor>5);  return izbor;  }  void main() {  int izbor = 0;  char \* tekst = NULL;  do {  cout << crt << "\t\t::TEKST EDITOR::";  izbor = PrikaziMeni();//pozivamo funkciju za prikaz menija  //korisnika onemogucavamo da izvrsi operacije nad tekstom koji  //koji jos uvijek nije unesen  if (tekst == NULL && (izbor == 2 || izbor == 3 || izbor == 4))  cout << crt << "Prvo unesite tekst!" << crt;  else {  switch (izbor) {  case 1:  UnosTeksta(tekst); break;  case 2:  DodajTekst(tekst); break;  case 3:  Informacije(tekst); break;  case 4:  Pretraga(tekst); break;  }  }  system("pause>0"); system("cls");  } while (izbor != 5); if (tekst != NULL)  OslobodiMemoriju(tekst);  cout << crt;  } |

# Zadatak 13 – Pametni pokazivači

Prepravite 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>  using namespace std;  //----------struktura DATUM-------------  struct Datum  {  int d;  int m;  int g;  };  void info(Datum\* date)  {  if (date == nullptr)  return;    cout << (\*date).d << "." << date->m << "." << date->g;  }  Datum\* ucitaj\_datum()  {  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(Osoba\* o)  {  if (o == nullptr)  return;  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);  //ili  //info(&osobe[i]);  //ili  //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;  //ili  // 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 = new PrijavaZaIspit[max\_p]; //umjesto PrijavaZaIspit prijave[max\_p];  };  Ispit\* dodaj\_ispit()  {  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(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;  }  Ispit\* odaberite\_ispit(Ispit\* ispiti, int max)  {  for (int i = 0; i<max; i++)  {  cout << i + 1 << ": ";  info(ispiti + i);  //ili  //info(&ispiti[i]);  }  cout << "Unesite RB ispita: ";  int x;  cin >> x;  return &ispiti[x - 1];  //ili  //return ispiti + x - 1;  }  void main()  {  const int max\_ispiti = 10;  Ispit\* ispiti = new 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);  //ili  //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++)  {  Osoba\* s = i->prijave[j].student;  Datum\* dP = i->prijave[j].datum\_prijave;  Datum\* dO = i->prijave[j].datum\_odjave;  info(s);  cout << " Datum prijave: ";  info(dP);  cout << " Datum odjave: ";  info(dO);  cout << endl;  }  }  if (x == 4)  {  Ispit\* i = odaberite\_ispit(ispiti, brojac\_ispita);  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)  {  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);  } |

|  |
| --- |
| #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);  } |

Autor:

mr. Adil Joldić

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