const char\* PORUKA = "\n-------------------------------------------------------------------------------\n"

"0. PROVJERITE DA LI PREUZETI ZADACI PRIPADAJU VASOJ GRUPI (G1/G2)\n"

"1. SVE KLASE TREBAJU POSJEDOVATI ADEKVATAN DESTRUKTOR\n"

"2. NAMJERNO IZOSTAVLJANJE KOMPLETNIH I/ILI POJEDINIH DIJELOVA DESTRUKTORA CE BITI OZNACENO KAO TM\n"

"3. SPASAVAJTE PROJEKAT KAKO BI SE SPRIJECILO GUBLJENJE URADJENOG ZADATKA\n"

"4. ATRIBUTI, NAZIVI FUNKCIJA, TE BROJ I TIP PARAMETARA MORAJU BITI IDENTICNI ONIMA KOJI SU KORISTENI U TESTNOM CODE-U,\n"

"\tOSIM U SLUCAJU DA POSTOJI ADEKVATAN RAZLOG ZA NJIHOVU MODIFIKACIJU. OSTALE\n"

"\tPOMOCNE FUNKCIJE MOZETE IMENOVATI I DODAVATI PO ZELJI.\n"

"5. IZUZETAK BACITE SAMO U FUNKCIJAMA U KOJIMA JE TO NAZNACENO.\n"

"6. FUNKCIJE KOJE NE IMPLEMENTIRATE TREBAJU BITI OBRISANE (KAKO POZIV TAKO I DEFINICIJA)!\n"

"7. NA KRAJU ISPITA SVOJE RJESENJE KOPIRATE U .DOCX FAJL (IMENOVAN BROJEM INDEKSA)!\n"

"8. RJESENJA ZADATKA POSTAVITE NA FTP SERVER U ODGOVARAJUCI FOLDER!\n"

"9. NEMOJTE POSTAVLJATI VISUAL STUDIO PROJEKTE, VEC SAMO .DOCX FAJL SA VASIM RJESENJEM!\n"

"-------------------------------------------------------------------------------\n";

const char\* crt = "\n-------------------------------------------\n";

enum Predmet { UIT, PRI, PRII, PRIII, RSI, RSII };

enum Dupliranje { SA\_DUPLIKATIMA, BEZ\_DUPLIKATA };

const char\* NIJE\_VALIDNA = "<VRIJEDNOST\_NIJE\_VALIDNA>";

char\* GetNizKaraktera(const char\* sadrzaj, bool dealociraj = false) {

if (sadrzaj == nullptr)return nullptr;

int vel = strlen(sadrzaj) + 1;

char\* temp = new char[vel];

strcpy\_s(temp, vel, sadrzaj);

if (dealociraj)

delete[]sadrzaj;

return temp;

}

template<class T1, class T2>

class Kolekcija {

T1\* \_elementi1;

T2\* \_elementi2;

int\* \_trenutno;

Dupliranje \_dupliranje;

public:

Kolekcija(Dupliranje dupliranje = SA\_DUPLIKATIMA) {

\_trenutno = nullptr;

\_elementi1 = nullptr;

\_elementi2 = nullptr;

\_dupliranje = dupliranje;

}

~Kolekcija() {

delete \_trenutno; \_trenutno = nullptr;

delete[] \_elementi1; \_elementi1 = nullptr;

delete[] \_elementi2; \_elementi2 = nullptr;

}

T1 getElement1(int lokacija)const { return \_elementi1[lokacija]; }

T2 getElement2(int lokacija)const { return \_elementi2[lokacija]; }

int getTrenutno()const { return \*\_trenutno; }

friend ostream& operator<< (ostream& COUT, const Kolekcija& obj) {

for (size\_t i = 0; i < \*obj.\_trenutno; i++)

COUT << obj.getElement1(i) << " " << obj.getElement2(i) << endl;

return COUT;

}

};

class Datum {

int\* \_dan, \* \_mjesec, \* \_godina;

public:

Datum(int dan = 1, int mjesec = 1, int godina = 2000) {

\_dan = new int(dan);

\_mjesec = new int(mjesec);

\_godina = new int(godina);

}

~Datum() {

delete \_dan; \_dan = nullptr;

delete \_mjesec; \_mjesec = nullptr;

delete \_godina; \_godina = nullptr;

}

friend ostream& operator<< (ostream& COUT, const Datum& obj) {

COUT << \*obj.\_dan << "." << \*obj.\_mjesec << "." << \*obj.\_godina;

return COUT;

}

Datum(const Datum& obj) {

\_dan = new int(\*obj.\_dan);

\_mjesec = new int(\*obj.\_mjesec);

\_godina = new int(\*obj.\_godina);

}

};

class Pitanje {

char\* \_sadrzaj;

//int se odnosi na ocjenu u opsegu 1 – 5, a Datum na datum kada je odgovor/rjesenje ocijenjeno

Kolekcija<int, Datum\*> \_ocjeneRjesenja;

public:

Pitanje(const char\* sadrzaj = "") {

\_sadrzaj = GetNizKaraktera(sadrzaj);

}

~Pitanje() {

delete[] \_sadrzaj; \_sadrzaj = nullptr;

}

char\* GetSadrzaj() { return \_sadrzaj; }

Kolekcija<int, Datum\*>& GetOcjene() { return \_ocjeneRjesenja; }

};

class Ispit {

Predmet \_predmet;

//string se odnosi na napomenu/zapazanje nastavnika

Kolekcija<Pitanje, string> \_pitanjaOdgovori;

public:

Ispit(Predmet predmet = UIT) {

\_predmet = predmet;

}

Kolekcija<Pitanje, string>& GetPitanjaOdgovore() { return \_pitanjaOdgovori; }

Predmet GetPredmet() { return \_predmet; }

friend ostream& operator<< (ostream& COUT, const Ispit& obj) {

COUT << obj.\_predmet << endl;

for (size\_t i = 0; i < obj.\_pitanjaOdgovori.getTrenutno(); i++)

COUT << obj.\_pitanjaOdgovori;

return COUT;

}

};

class Korisnik {

char\* \_imePrezime;

string \_emailAdresa;

string \_lozinka;

public:

Korisnik(const char\* imePrezime, string emailAdresa, string lozinka)

{

\_imePrezime = GetNizKaraktera(imePrezime);

\_emailAdresa = emailAdresa;

\_lozinka = ValidirajLozinku(lozinka) ? lozinka : NIJE\_VALIDNA;

}

~Korisnik() { delete[] \_imePrezime; \_imePrezime = nullptr; }

string GetEmail() { return \_emailAdresa; }

string GetLozinka() { return \_lozinka; }

char\* GetImePrezime() { return \_imePrezime; }

virtual void Info() = 0;

};

class Kandidat : public Korisnik {

vector<Ispit\*> \_polozeniPredmeti;

public:

Kandidat(const char\* imePrezime, string emailAdresa, string lozinka) : Korisnik(imePrezime, emailAdresa, lozinka) {

}

~Kandidat() {

for (size\_t i = 0; i < \_polozeniPredmeti.size(); i++)

delete \_polozeniPredmeti[i];

}

friend ostream& operator<< (ostream& COUT, Kandidat& obj) {

COUT << obj.GetImePrezime() << " " << obj.GetEmail() << " " << obj.GetLozinka() << endl;

for (size\_t i = 0; i < obj.\_polozeniPredmeti.size(); i++)

COUT << obj.\_polozeniPredmeti[i];

return COUT;

}

vector<Ispit\*>& GetPolozeniPredmeti() { return \_polozeniPredmeti; }

};

const char\* GetOdgovorNaPrvoPitanje() {

cout << "Pitanje -> Pojasnite na koji nacin tip nasljedjivanja (javno, zasticeno, privatno) utjece na dostupnost atributa bazne klase unutar izvedene klase?\n";

return "Odgovor -> OVDJE UNESITE VAS ODGOVOR";

}

const char\* GetOdgovorNaDrugoPitanje() {

cout << "Pitanje -> Ukratko pojasnite primjenu funkcija seekg i tellg?\n";

return "Odgovor -> OVDJE UNESITE VAS ODGOVOR";

}

void main() {

cout << PORUKA;

cin.get();

cout << GetOdgovorNaPrvoPitanje() << endl;

cin.get();

cout << GetOdgovorNaDrugoPitanje() << endl;

cin.get();

Datum datum19062020(19, 6, 2020),

datum20062020(20, 6, 2020),

datum30062020(30, 6, 2020),

datum05072020(5, 7, 2020);

int kolekcijaTestSize = 10;

Kolekcija<int, int> kolekcija1(BEZ\_DUPLIKATA);

for (int i = 0; i < kolekcijaTestSize; i++)

kolekcija1.AddElement(i, i);

cout << kolekcija1 << endl;

try {

/\*ukoliko dupliranje vrijednosti nije dozvoljeno (BEZ\_DUPLIKATA)

metoda AddElement baca izuzetak u slucaju da se pokusa dodati par sa vrijednostima

identicnim postojecem paru unutar kolekcije \*/

kolekcija1.AddElement(3, 3);

}

catch (exception& err) {

cout << crt << "Greska -> " << err.what() << crt;

}

cout << kolekcija1 << crt;

Kolekcija<int, int> kolekcija2;

kolekcija2 = kolekcija1;

cout << kolekcija2 << crt;

/\*na osnovu vrijednosti tip T1 (u primjeru vrijednost 1) pronalazi i vraca iz kolekcije sve parove

koji kao vrijednost T1 imaju proslijedjenu vrijednost\*/

cout << kolekcija1[1] << crt;

/\* npr. ako unutar kolekcije postoje parovi:

0 0

1 9

1 1

2 2

3 3

ispis dobijenih/vracenih vrijednosti ce biti sljedeci:

1 9

1 1

\*/

Pitanje sortiranjeNiza("Navedite algoritme za sortiranje clanova niza."),

dinamickaMemorija("Navedite pristupe za upravljanje dinamickom memorijom."),

visenitnoProgramiranje("Na koji se sve nacin moze koristiti veci broj niti tokom izvrsenja programa."),

regex("Navedite par primjera regex validacije podataka.");

/\*svako pitanje moze imati vise ocjena tj. razlicita rjesenja/odgovori se mogu postaviti u vise navrata.

- razmak izmedju postavljanja dva rjesenja mora biti najmanje 3 dana

- nije dozvoljeno dodati ocjenu sa ranijim datumom u odnosu na vec evidentirane (bez obzira sto je stariji od 3 dana)

\*/

if (sortiranjeNiza.AddOcjena(1, datum19062020))

cout << "Ocjena evidentirana!" << endl;

if (!sortiranjeNiza.AddOcjena(5, datum20062020))

cout << "Ocjena NIJE evidentirana!" << endl;

if (sortiranjeNiza.AddOcjena(5, datum30062020))

cout << "Ocjena evidentirana!" << endl;

// ispisuje sadrzaj/tekst pitanja, ocjene (zajedno sa datumom) i prosjecnu ocjenu za sve odgovore/rjesenja

// ukoliko pitanje nema niti jednu ocjenu prosjecna treba biti 0

cout << sortiranjeNiza << endl;

if (ValidirajLozinku("\*2gT2x"))

cout << "Lozinka validna" << endl;

if (ValidirajLozinku("\*7aT2x"))

cout << "Lozinka validna" << endl;

if (ValidirajLozinku("\_6gU9z"))

cout << "Lozinka validna" << endl;

if (ValidirajLozinku("\*3aB1y"))

cout << "Lozinka validna" << endl;

if (ValidirajLozinku("\*1a T2l"))

cout << "Lozinka validna" << endl;

if (!ValidirajLozinku("-1a T2l"))

cout << "Lozinka NIJE validna" << endl;

/\*

za autentifikaciju svaki korisnik mora posjedovati lozinku koja sadrzi 6 ili 7 znakova postujuci sljedeca pravila:

- pocinje sa znak \* (zvjezdica) ili \_ (donja crtica), nakon cega slijedi

- broj, nakon cega slijedi

- malo slovo, nakon cega slijedi

- razmak koji NIJE OBAVEZAN, nakon cega slijedi

- veliko slovo, nakon cega slijedi

- broj, nakon cega slijedi

- malo slovo.

za provjeru validnosti lozinke koristiti globalnu funkciju ValidirajLozinku, a unutar nje regex metode.

validacija lozinke se vrsi unutar konstruktora klase Korisnik, a u slucaju da nije validna

postaviti je na podrazumijevanu vrijednost: <VRIJEDNOST\_NIJE\_VALIDNA>

\*/

Korisnik\* jasmin = new Kandidat("Jasmin Azemovic", "jasmin@kursevi.ba", "\*2gT2x");

Korisnik\* adel = new Kandidat("Adel Handzic", "adel@edu.kursevi.ba", "\_6gU9z");

Korisnik\* lozinkaNijeValidna = new Kandidat("John Doe", "john.doe@google.com", "johndoe");

/\*

svi odgovori na nivou jednog predmeta (PRI, PRII... ) se evidentiraju unutar istog objekta tipa Ispit tj. pripadajuceg objekta tipa Pitanje,

tom prilikom onemoguciti:

- dodavanje istih (moraju biti identicne vrijednosti svih atributa) odgovora na nivou jednog predmeta,

- dodavanje odgovora za visi predmet ako prethodni predmet nema evidentirana najmanje 3 pitanja ili nema prosjecnu ocjenu svih pitanja vecu od 3.5

(onemoguciti dodavanje pitanja za PRII ako ne postoje najmanje 3 pitanja za predmet PRI ili njihov prosjek nije veci od 3.5)

funkcija vraca true ili false u zavisnosti od (ne)uspjesnost izvrsenja

\*/

//ukoliko je potrebno, doraditi klase da nacin da omoguce izvrsenje naredne linije koda

Kandidat\* jasminPolaznik = dynamic\_cast<Kandidat\*>(jasmin);

if (jasminPolaznik != nullptr) {

if (jasminPolaznik->AddPitanje(PRI, dinamickaMemorija, "nedostaje operator delete"))

cout << "Pitanje uspjesno dodano!" << crt;

//ne treba dodati visenitnoProgramiranje jer ne postoje evidentirana 3 pitanja za predmet PRI

if (!jasminPolaznik->AddPitanje(PRII, visenitnoProgramiranje))

cout << "Pitanje NIJE uspjesno dodano!" << crt;

if (jasminPolaznik->AddPitanje(PRI, visenitnoProgramiranje))

cout << "Pitanje uspjesno dodano!" << crt;

if (jasminPolaznik->AddPitanje(PRI, regex, "unutar posljednjeg dijela nedostaje opis glavnih operatora"))

cout << "Pitanje uspjesno dodano!" << crt;

if (jasminPolaznik->AddPitanje(PRI, sortiranjeNiza))

cout << "Pitanje uspjesno dodano!" << crt;

//ne treba dodati sortiranjeNiza jer je vec dodana za predmet PRI

if (!jasminPolaznik->AddPitanje(PRI, sortiranjeNiza))

cout << "Pitanje NIJE uspjesno dodano!" << crt;

//ispisuje sve dostupne podatke o kandidatu

cout << \*jasminPolaznik << crt;

//vraca broj ponavljanja odredjene rijeci unutar napomena nastalih tokom polaganja ispita.

int brojPonavljanja = (\*jasminPolaznik)("nedostaje");

cout << "Rijec nedostaje se ponavlja " << brojPonavljanja << " puta." << endl;

}

/\*nakon evidentiranja ocjene na bilo kojem odgovoru, kandidatu se salje email sa porukom:

FROM:info@kursevi.ba

TO: emailKorisnika

Postovani ime i prezime, evidentirana vam je ocjena X za odgovor na pitanje Y. Dosadasnji uspjeh (prosjek ocjena)

za pitanje Y iznosi F, a ukupni uspjeh (prosjek ocjena) na svim predmetima iznosi Z.

Pozdrav.

EDUTeam.

slanje email poruka implemenitrati koristeci zasebne thread-ove.

\*/

delete jasmin;

delete adel;

delete lozinkaNijeValidna;

cin.get();

system("pause>0");

}