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. 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 Pojas { BIJELI, ZUTI, NARANDZASTI, ZELENI, PLAVI, SMEDJI, CRNI };

const int brojTehnika = 6;

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, int max = 10>

class Kolekcija {

T1\* \_elementi1[max] = { nullptr };

T2\* \_elementi2[max] = { nullptr };

int \_trenutno;

public:

Kolekcija() { \_trenutno = 0; }

~Kolekcija() {

for (size\_t i = 0; i < \_trenutno; i++) {

delete \_elementi1[i]; \_elementi1[i] = nullptr;

delete \_elementi2[i]; \_elementi2[i] = nullptr;

}

}

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

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

int getTrenutno() { 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;

}

};

class Tehnika {

char\* \_naziv;

//int se odnosi na ocjenu u opsegu od 1 – 5, a Datum na datum kada je ocijenjena odredjena tehnika

Kolekcija<int, Datum, brojTehnika>\* \_ocjene;

public:

Tehnika(const char\* naziv) {

\_naziv = GetNizKaraktera(naziv);

\_ocjene = new Kolekcija<int, Datum, brojTehnika>();

}

~Tehnika() {

delete[] \_naziv; \_naziv = nullptr;

delete \_ocjene; \_ocjene = nullptr;

}

char\* GetNaziv() { return \_naziv; }

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

};

class Polaganje {

Pojas \_pojas;

vector<Tehnika\*> \_polozeneTehnike;

public:

Polaganje(Pojas pojas = BIJELI) {

\_pojas = pojas;

}

~Polaganje() {

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

delete \_polozeneTehnike[i];

\_polozeneTehnike[i] = nullptr;

}

}

vector<Tehnika\*>& GetTehnike() { return \_polozeneTehnike; }

Pojas GetPojas() { return \_pojas; }

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

COUT << obj.\_pojas << endl;

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

COUT << \*obj.\_polozeneTehnike[i];

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; }

};

class KaratePolaznik {

vector<Polaganje> \_polozeniPojasevi;

public:

KaratePolaznik(const char\* imePrezime, string emailAdresa, string lozinka){

}

~KaratePolaznik() {

cout << crt << "DESTRUKTOR -> KaratePolaznik" << crt;

}

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

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

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

COUT << obj.\_polozeniPojasevi[i];

return COUT;

}

vector<Polaganje>& GetPolozeniPojasevi() { return \_polozeniPojasevi; }

};

const char\* GetOdgovorNaPrvoPitanje() {

cout << "Pitanje -> Pojasnite ulogu operatora reinterpret\_cast.\n";

return "Odgovor -> OVDJE UNESITE VAS ODGOVOR";

}

const char\* GetOdgovorNaDrugoPitanje() {

cout << "Pitanje -> Ukratko opisite znacaj i vrste pametnih pokazivaca.\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;

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

kolekcija1.AddElement(i, i);

cout << kolekcija1 << endl;

try {

/\*metoda AddElement baca izuzetak u slucaju da se pokusa

dodati vise od maksimalnog broja elemenata\*/

kolekcija1.AddElement(11, 11);

}

catch (exception & err) {

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

}

cout << kolekcija1 << crt;

kolekcija1.RemoveAt(2);

/\*uklanja par (T1 i T2) iz kolekcije koji se nalazi na lokaciji sa proslijedjenim indeksom.

nakon uklanjanja vrijednosti onemoguciti pojavu praznog prostora unutar kolekcije tj.

pomjeriti sve elemente koji se nalaze nakon proslijedjene lokacije za jedno mjesto unazad

npr. ako unutar kolekcije postoje elementi

0 0

1 1

2 2

3 3

nakon uklanjanja vrijednosti na lokaciji 1, sadrzaj kolekcije ce biti sljedeci

0 0

2 2

3 3

\*/

cout << kolekcija1 << crt;

kolekcija1.AddElement(9, 9, 2);

/\*funkciji AddElement se, kao treci parametar, moze proslijediti i lokacija na koju se dodaju

nove vrijednosti pri cemu treba zadrzati postojece vrijednosti pomjerene za jedno mjesto unaprijed

u odnosu na definisanu lokaciju npr. ako unutar kolekcije postoje elementi

0 0

1 1

2 2

3 3

nakon dodavanja vrijednosti 9 i 9 na lokaciju 1, sadrzaj kolekcije ce biti sljedeci

0 0

9 9

1 1

2 2

3 3

\*/

cout << kolekcija1 << crt;

Kolekcija<int, int> kolekcija2 = kolekcija1;

cout << kolekcija1 << crt;

//na osnovu vrijednosti T1 mijenja vrijednost T2.

kolekcija1[9] = 2;

/\* npr.ako unutar kolekcije postoje elementi:

0 0

9 9

1 1

2 2

3 3

nakon promjene vrijednosti sadrzaj kolekcije ce biti sljedeci

0 0

9 2

1 1

2 2

3 3

\*/

Tehnika choku\_zuki("choku\_zuki"),

gyaku\_zuki("gyaku\_zuki"),

kizami\_zuki("kizami\_zuki"),

oi\_zuki("oi\_zuki");

/\*svaka tehnika moze imati vise ocjena tj. moze se polagati u vise navrata.

- razmak izmedju polaganja dvije tehnike 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 (choku\_zuki.AddOcjena(1, datum19062020))

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

if (!choku\_zuki.AddOcjena(5, datum20062020))

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

if (choku\_zuki.AddOcjena(5, datum30062020))

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

/\* ispisuje: naziv tehnike, ocjene (zajedno sa datumom) i prosjecnu ocjenu za tu tehniku

ukoliko tehnika nema niti jednu ocjenu prosjecna treba biti 0\*/

cout << choku\_zuki << endl;

if (ValidirajLozinku("john4Do\*e"))

cout << "OK" << crt;

if (!ValidirajLozinku("john4Doe"))

cout << "Specijalni znak?" << crt;

if (!ValidirajLozinku("jo\*4Da"))

cout << "7 znakova?" << crt;

if (!ValidirajLozinku("4jo-hnoe"))

cout << "Veliko slovo?" << crt;

if (ValidirajLozinku("@john2Doe"))

cout << "OK" << crt;

/\*

za autentifikaciju svaki korisnik mora posjedovati lozinku koja sadrzi:

- najmanje 7 znakova

- velika i mala slova

- najmanje jedan broj

- najmanje jedan specijalni znak

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 KaratePolaznik("Jasmin Azemovic", "jasmin@karate.ba", "j@sm1N\*");

Korisnik\* adel = new KaratePolaznik("Adel Handzic", "adel@edu.karate.ba", "4Ade1\*H");

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

/\*

sve tehnike na nivou jednog pojasa (ZUTI, ZELENI ... ) se evidentiraju unutar istog objekta tipa Polaganje,

tom prilikom onemoguciti:

- dodavanje istih (moraju biti identicne vrijednosti svih atributa) tehnika na nivou jednog pojasa,

- dodavanje tehnika za visi pojas ako prethodni pojas nema evidentirane najmanje 3 tehnike ili nema prosjecnu ocjenu svih tehnika vecu od 3.5

(onemoguciti dodavanje tehnike za NARANDZASTI ako ne postoji najmanje 3 tehnike za ZUTI pojas ili njihov prosjek nije veci od 3.5)

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

\*/

//doraditi klase da nacin da omoguce izvrsenje naredne linije koda

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

if (jasminPolaznik != nullptr) {

if (jasminPolaznik->AddTehniku(ZUTI, gyaku\_zuki))

cout << "Tehnika uspjesno dodan!" << crt;

//ne treba dodati kizami\_zuki jer ne postoje evidentirane 3 tehnike za ZUTI pojas

if (!jasminPolaznik->AddTehniku(NARANDZASTI, kizami\_zuki))

cout << "Tehnika NIJE uspjesno dodana!" << crt;

if (jasminPolaznik->AddTehniku(ZUTI, choku\_zuki))

cout << "Tehnika uspjesno dodan!" << crt;

//ne treba dodati choku\_zuki jer je vec dodana za zuti pojas

if (!jasminPolaznik->AddTehniku(ZUTI, choku\_zuki))

cout << "Tehnika NIJE uspjesno dodana!" << crt;

//ispisuje sve dostupne podatke o karate polazniku

cout << \*jasminPolaznik << crt;

}

/\*nakon evidentiranja tehnike na bilo kojem pojasu kandidatu se salje email sa porukom:

FROM:info@karate.ba

TO: emailKorisnika

Postovani ime i prezime, evidentirana vam je thenika X za Y pojas. Dosadasnji uspjeh (prosjek ocjena)

na pojasu Y iznosi F, a ukupni uspjeh (prosjek ocjena) na svim pojasevima iznosi Z.

Pozdrav.

KARATE Team.

slanje email poruka implemenitrati koristeci zasebne thread-ove.

\*/

//osigurati da se u narednim linijama poziva i destruktor klase KaratePolaznik

delete jasmin;

delete adel;

delete emailNijeValidan;

cin.get();

system("pause>0");

}