ZAVRŠNI PROJEKAT IZ PREDMETA NAPREDNE TEHNIKE PROGRAMIRANJA, AKADEMSKA 2020./2021.GODINA, NA TEMU:

AUTOBUSKA STANICA

Odsjek: Softversko inženjerstvo

Naziv predmeta: Napredne tehnike programiranja

Asistent: Edin Tabak

Semestar: Ljetni

Imena i prezimena studenata: Kerim Žuna, Semra Kermo, Adrian Jurišić

Brojevi indeksa: 199, 172, 171

Zenica, juni 2021.godine

OBAVEZNI ELEMENTI PROJEKTA

1. ENUMERACIJE su tipovi podataka koji mogu biti definisani od strane programera. Glavni zadatak enumeracija jeste dodjeljivanje imena odnosno stringova integralnim konstantama, kako bi sam kod bio što pregledniji.

```
enum Starost {Dijete=1, Student, Odrasli, Penzioner};
enum PovratnaKarta {Jednosmjerna, Povratna};
enum Legenda {SvakiDan=7, Praznik=6, Raspust=5};
```

- **Enum "Starost"** je iskorištena prilikom rezervacije nove karte u programu. Cijena karte ovisi od starosti osobe koja rezerviše tu kartu.

```
double CijenaKarte;
switch (x){
    case 1:
        UzrastPutnika = "Dijete";
        CijenaKarte = (Podaci[niz[IzborDatuma]].Cijena)*0.65;
        break;
    case 2:
        UzrastPutnika = "Student";
        CijenaKarte = (Podaci[niz[IzborDatuma]].Cijena)*0.9;
        break;
    case 3:
        UzrastPutnika = "Odrasli";
        CijenaKarte = Podaci[niz[IzborDatuma]].Cijena;
        break;
    case 4:
        UzrastPutnika = "Penzioner";
        CijenaKarte = (Podaci[niz[IzborDatuma]].Cijena)*0.85;
        break;
```

- **Enum "Povratna karta"** je također iskorištena prilikom rezervacije nove karte u programu. Cijena karte ovisi od toga da li je jednosmjerna ili povratna.

```
string Povratna;
switch (y){
    case 0:
        Povratna = "Jednosmjerna";
        CijenaKarte = CijenaKarte*0.65;
        break;
    case 1:
        Povratna = "Povratna";
        break;
}
```

- **Enum "Legenda"** je iskorištena prilikom određivanja učestalosti određenih autobuskih linija. Naprimjer, neke linije su aktivne svaki dan (7 dana u sedmici), neke liniji su aktivne svaki dan osim praznicima. Neradni praznik obično traje 1 dan u sedmici, tako da je ta linija, pod nazivom "Praznik", aktivna samo 6 dana u sedmici. Na kraju, neke linije su aktivne samo radnim danima, odnosno od ponedjeljka do petka, naziv enumeracije "Raspust".

```
case 6:
    if(provjera == "SvakiDan") Lokalac[h].Dan = SvakiDan;
    else if(provjera == "Praznik") Lokalac[h].Dan = Praznik;
    else Lokalac[h].Dan = Raspust;
    break;
```

2. STRUKTURE: Struktura ustvari predstavlja grupu varijabli različitih tipova pod jednim imenom. Koriste se kako bi olakšali rad i pisanje samog koda pri komplikovanijim projektima. Također doprinose i boljem vizualnom izgledu koda. Za primjer možemo uzeti varijablu ***int godina*** koja označava brojčanu vrijednost godine. Može se desiti da u samom kodu tu varijablu moramo upotrebljavati više puta, od nje kreirati niz podataka odnosno, u ovom slučaju, niz koji sadrži više brojčanih vrijednosti godina. Definisanje strukture koja sadrži primjer već spomenute varijable, omogućava nam pozivanje te varijable i lahko mijenjanje njene vrijednosti, kao i upotrebu sa datotekama.

- **"Struct Datum"** se koristi pri određivanju datuma polazaka i dolazaka određenih autobuskih linija u samom programu.

```
struct Datum{
    int dan;
    int mjesec;
    int godina;
```

- **"Struct Vrijeme"** se koristi pri određivanju vremena polazaka i dolazaka određenih autobuskih linija u samom programu.

```
struct Vrijeme{
    int sat;
    int minuta;
```

- **"Struct Putnik"** se koristi za upis osnovnih informacija o putnicima, kao što su ime, prezime, kontakt telefon. Također se koriste informacije o uzrastu i vrsta karte koju žele pri svome putovanju.

```
struct Putnik{
    string ime;
    string prezime;
    string KontaktBroj;
    Starost uzrast;
    PovratnaKarta karta;
};
```

- **“Struct Autobus”** se koristi za osnovne informacije o autobusu odnosno, liniji kojom taj autobus putuje. Neke od tih informacija su mjesto polazka i povratka, datum i vrijeme polaska i povratka, ime vozača, ID autobusa itd.

```
struct Autobus{
    string PolazakMjesto;
    string PovratakMjesto;
    Datum DatumPolaska;
    Vrijeme PolazakVrijeme;
    Datum DatumPovratka;
    Vrijeme PovratakVrijeme;
    string Agencija;
    double Cijena;
    int BrojMjesta;
    int ID;
    string ImeVozaca;
    string PrezimeVozaca;
};
```

3. DINAMIČKO ALOCIRANJE MEMORIJE za razliku od statičkog, alocira memoriju u “heap-u”. Kada klasično deklarišemo varijablu ili niz npr. ***“int n”*** koji nam predstavlja neki broj, memorija se automatski alocira i delocira od strane računara. Međutim, kada koristimo dinamičko alociranje memorije, mi kao programi smo dužni naglasiti kada smo završili sa korištenjem te varijable i kada nam više ne treba komandom ***delete ().***

```
Autobus *Podaci = new Autobus [BrojLinija];
```

dinamički alociran niz

```
}
delete [] Podaci;
```

dinamički dislociran niz

4. FUNKCIJE U STRUKTURI: Funkcija predstavlja grupu izvršnih komadni pod jednim imenom. Koristi se za olakšano pisanje koda. Ako npr. trebamo sortirani više nizova. Nakon kreiranja svakog niza možemo pozvati funkciju koju smo kreirali samo za obavljanje sortiranja. Funkcije se deklarišu prije ***int main ()*** funkcije, u kojoj se zapravo pozivaju. Funkcije u strukturi

su ustvari obične funkcije smještene u strukturu, odnosno funkcije koje koriste ili vrše određene zadatke nad elementima strukture.

```
struct Datum{
    int dan;
    int mjesec;
    int godina;

    bool DatumUnos(Datum datum){
        if(datum.godina<1000 || datum.godina>10000) return false;
        if(datum.dan<1 || datum.dan>31) return false;
        switch (datum.mjesec){
            case 1: case 3: case 5: case 7: case 8: case 10: case 12:
                if(datum.dan>31 || datum.dan<1) return false;
                else return true;
                break;
            case 2:
                if(datum.godina%4!=0 || (datum.godina%4==0 && datum.godina%100==0 && datum.godina%400!=0)){
                    if(datum.dan>28 || datum.dan<1) return false;
                    else return true;
                }else{
                    if(datum.dan>29 || datum.dan<1) return false;
                    else return true;
                }
                break;
            case 4: case 6: case 9: case 11:
                if(datum.dan>30 || datum.dan<1) return false;
                else return true;
                break;
            default:
```

funkcija u strukturi "Datum" za određivanje validnosti datuma

```
struct Vrijeme{
    int sat;
    int minuta;

    bool VrijemeUnos(Vrijeme vrijeme){
        if(vrijeme.sat<0 || vrijeme.sat>23) return false;
        else{
            if(vrijeme.minuta<0 || vrijeme.minuta>59) return false;
            else return true;
        }
    }
    bool VrijemeProvjera(Vrijeme polazak, Vrijeme dolazak){
        if(polazak.sat==dolazak.sat || polazak.sat>dolazak.sat) return false;
        else if(polazak.sat<dolazak.sat) return true;
    }
};
```

funkcija u strukturi "Vrijeme" za određivanje validnosti vremena

5. FUNKCIJE VAN STRUKTURE: Nekoliko primjera funkcija van strukture.

```
void PretragaVozaca (Autobus Podaci[], string ime, string prezime, int i, int n, int br){
    if (i==n){
        if (br==0){
            cout << "\t\tUneseni vozac ne postoji ili nema zakazane voznje." << endl;
        }
    }else if((Podaci[i].ImeVozaca) == ime && (Podaci[i].PrezimeVozaca == prezime)){
        br++;
        cout << "\t\t" << br << ". " << Podaci[i].PolazakMjesto << "-" << Podaci[i].PovratakMjesto << endl;
        return PretragaVozaca (Podaci,ime,prezime,i+1,n,br);
    }else{
        return PretragaVozaca (Podaci,ime,prezime,i+1,n,br);
    }
}
```

funkcija za pretragu vozača

```
string setPass(bool show_asterisk = true){
    const char BACKSPACE = 8;
    const char ENTER = 13;
    string pass = "";
    char c = ' ';
    while ((c = _getch()) != ENTER){
        if (c == BACKSPACE){
            if (pass.length() != 0){
                if (show_asterisk)
                    cout << "\b \b";
                pass.resize(pass.length() - 1);
            }
        }else if (c == 0 || c == 224){
            _getch();
            continue;
        }else{
            pass.push_back(c);
            cout << '*';
        }
    }
    cout << endl;
    return pass;
}
```

funckija za sakrivanje passworda prilikom prijave

```
int BrSlobodnihMjesta(int ID, int BrSlobodnih){
    string line, pomocna;
    pomocna = to_string(ID);
    pomocna = pomocna + ".txt";
    ifstream Putnici(pomocna);
    if(Putnici.fail()){
        return BrSlobodnih;
    }else{
        while(getline(Putnici,line)){
            BrSlobodnih--;
        }
    }
    return BrSlobodnih;
}
```

funkcija za određivanje broja slobodnih mjesta u autobusu

```
bool Prijava (){
    string line, username, password, temp;
    do{
    cout<<"\t"<<setfill('-')<<setw(30)<<"-"<<endl;
    cout<<"\t| AUTOBUSKA STANICA U ZENICI |\n";
    cout<<"\t"<<setfill('-')<<setw(30)<<"-"<<endl;
    cout << "\t\n\t\tLOGIN\t" << endl;
    cout<<"\t"<<setfill('-')<<setw(20)<<"-"<<endl;
    cout << "\tUsername: ";
    getline (cin, username);
    cout << "\tPassword: ";
    password=setPass();
    ifstream Login ("Osoblje.txt");
    if(Login.fail()){
        cout<<"\t"<<setfill('-')<<setw(20)<<"-"<<endl;
        cout << "\n[ERROR] Datoteku nije moguce otvoriti!\n" << endl;
    }else{
        while(!Login.eof()){
            Login >> line;
            Login >> temp;
            if (line.compare(username)==0 && temp.compare(password)==0){
                return true;
            }
        }
    }
    cout<<"\t"<<setfill('-')<<setw(20)<<"-"<<endl;
    cout << "\n[ERROR]\tPogresan username ili password!\n\tPokusajte ponovo.\n" << endl;
    system ("pause");
    system ("cls");
    }while(true);
```

funkcija za prijavu u program

6. REKURZIVNA FUNKCIJA predstavlja funkciju koja, ukoliko se neki uslov ispuni ili ne ispuni, poziva samu sebe. Ovaj tip funkcije, iako često neiskorišten, u nekim slučajevima može da bude praktičniji i jednostavniji za primjenu u odnosu na standardni oblik funkcije.

```
void PretragaVozaca (Autobus Podaci[], string ime, string prezime, int i, int n, int br){
    if (i==n){
        if (br==0){
            cout << "\t\tUneseni vozac ne postoji ili nema zakazane voznje." << endl;
        }
    }else if((Podaci[i].ImeVozaca) == ime && (Podaci[i].PrezimeVozaca == prezime)){
        br++;
        cout << "\t\t" << br << ". " << Podaci[i].PolazakMjesto << "-" << Podaci[i].PovratakMjesto << endl;
        return PretragaVozaca (Podaci,ime,prezime,i+1,n,br);
    }else{
        return PretragaVozaca (Podaci,ime,prezime,i+1,n,br);
    }
}
```

7. RAD SA DATOTEKAMA podrazumijeva upisivanje podataka u datoteku ili ispisivanje podataka iz datoteke koja već sadrži sve potrebne informacije za naš program. Najprostiji primjer ovoga jeste ispisivanje ID broja autobusa, njegove linije, datuma i vremena polaska i dolaska, cijenu karte, broja sjedišta, imena vozača itd. U nastavku slijedi jedan od primjera:

```
string line;
ifstream Autobusi("Autobusi.txt");
if(Autobusi.fail()){
    cout << "\tDatoteka nije pronadjena!\n\t";
    system("pause");
    system("cls");
    Meni();
}
while(!Autobusi.eof()){
    getline(Autobusi,line);
    BrojLinija++;
}
Autobusi.close();
Autobus *Podaci = new Autobus [BrojLinija];
int k=0;
Autobusi.open("Autobusi.txt");
while(!Autobusi.eof()){
    string l = "";
    getline(Autobusi,line);
    int i=0, br=0;
    string provjera = "";
    do{
        if(line[i] == '/'){
            br++;
```

primjer rada sa datotekom, čitanje podataka iz datoteke i upis tih podataka u strukturu

8. RAD SA KARAKTERIMA:

```
case 3:
    l = "";
    l = l + provjera[0] + provjera[1];
    Podaci[k].PolazakVrijeme.sat = stoi(l);
    l = "";
    l = l + provjera[3] + provjera[4];
    Podaci[k].PolazakVrijeme.minuta = stoi(l);
    break;
```

pretvaranje brojevnih vrijednosti u string (int → string),

9. MENI – kreiranje i korištenje menija, što u suštini podrazumijeva pružanje određenog broja opcija ili mogućnosti korisniku ovog programa. Korisnik ima pravo da izabere koju opciju od ponuđenih želi i na osnovu njegovog odabira izvršavaju se određene radnje i operacije.

```
int Meni1(){
    int izbor;
    cout<<"\t"<<setfill('-')<<setw(30)<<"-"<<endl;
    cout<<"\t| AUTOBUSKA STANICA U ZENICI |\n";
    cout<<"\t"<<setfill('-')<<setw(30)<<"-"<<endl;
    cout << "\n\t\tMENI\n";
    cout<<"\t"<<setfill('-')<<setw(20)<<"-";
    cout << "\n\t1. Gradske linije\n";
    cout << "\t2. Vangradske linije\n";
    cout << "\t3. Log Out\n";
    cout << "\t4. Iskljuci program" << endl;
    cout<<"\t"<<setfill('-')<<setw(20)<<"-";
    cout << "\n\tVas izbor: ";
```

početni meni

```
int Meni2 (){
    int izbor;
    cout<<"\t"<<setfill('-')<<setw(30)<<"-"<<endl;
    cout<<"\t| AUTOBUSKA STANICA U ZENICI |\n";
    cout<<"\t"<<setfill('-')<<setw(30)<<"-"<<endl;
    cout << "\n\t\tGRADSKE LINIJE\n";
    cout<<"\t"<<setfill('-')<<setw(30)<<"-"<<endl;
    cout << "\t1. Ispis svih linija" << endl;
    cout << "\t2. Pretraga linija" << endl;
    cout << "\t3. Izlaz" << endl;
    cout<<"\t"<<setfill('-')<<setw(30)<<"-"<<endl;
    cout << "\tVas izbor: ";
    cin >> izbor;
    switch (izbor){
```

meni za gradske linije

```
int Meni3 (){
    cout<<"\t"<<setfill('-')<<setw(30)<<"-"<<endl;
    cout<<"\t| AUTOBUSKA STANICA U ZENICI |\n";
    cout<<"\t"<<setfill('-')<<setw(30)<<"-"<<endl;
    cout << "\n\t\tVANGRADSKE LINIJE\n";
    cout<<"\t"<<setfill('-')<<setw(30)<<"-"<<endl;
    int izbor;
    cout << "\t1. Rezervacije" << endl;
    cout << "\t2. Pretraga linija" << endl;
    cout << "\t3. Unos novih linija" << endl;
    cout << "\t4. Otkazivanje rezervacija" << endl;
    cout << "\t5. Izlaz" << endl;
    cout<<"\t"<<setfill('-')<<setw(30)<<"-"<<endl;
    cout << "\tVas izbor: ";
```

meni za vangradske linije

```
int MeniPretraga (){
    cout<<"\t"<<setfill('-')<<setw(30)<<"-"<<endl;
    cout<<"\t| AUTOBUSKA STANICA U ZENICI |\n";
    cout<<"\t"<<setfill('-')<<setw(30)<<"-"<<endl;
    cout << "\n\t     KATEGORIJE PRETRAGE\n";
    cout<<"\t"<<setfill('-')<<setw(30)<<"-"<<endl;
    int izbor;
    cout << "\t1. Agencija" << endl;
    cout << "\t2. Destinacija" << endl;
    cout << "\t3. ID" << endl;
    cout << "\t4. Datum polaska" <<endl;
    cout << "\t5. Vozac" << endl;
    cout << "\t6. Izlaz" << endl;
    cout<<"\t"<<setfill('-')<<setw(30)<<"-"<<endl;
    cout << "\tVas izbor: ";
```

meni za pretragu

KRATKI SAŽETAK PROJEKTA

Projekat je pravljen iz perspektive radnika/ce u autobuskoj stanici. Pružili smo mogućnost prijave sa korisničkim imenom i šifrom, pretpostavljajući da, naravno, postoji više različitih radnika. Prijavom smo, teoretski gledajući, omogućili nadležnima u autobuskoj stanici da prate rad svojih radnika. Radnik/ca imaju mogućnost rezervisanja ili otkazivanja rezervacije za lokalne odnosno gradske, kao i za vangradske odnosno međunarodne linije. Također, imaju uvid u sve postojeće rezervacije, cijene karata, autobuse i njihove linije, kao i vozače tih autobusa, brojeve mjesta u autobusima. Pored uvida u dosadašnje postojeće linije, radnik/ca ima mogućnost da unese novu liniju, npr. ***Zenica – Donji Vakuf*** koja do sada nije postojala. Omogućili smo ispis grešaka za sve nevalidne datume polazaka ili dolazaka, kao i vremena istih, npr. datum dolaska kalendarski ne može prethoditi datum polaska. Za vrijeme dolaska i polaska važi isto. Pretraga je omogućena prema raznim vrijednostima, kao što su ID autobusa, agencija koja tu autobusku liniju pruža, destinacija, datum polaska, ime vozača.