

Preliminarūs 2024 metų informacinių technologijų egzamino atsakymai

IT VBE preliminarūs atsakymai

Saugus ir teisėtas informacijos ir interneto naudojimas

1 C.

2. A.

3. Naudojamas http, o ne saugesnis https šifravimas.

4. Antivirusinė programa, nuolat tikrinanti programų ir diskų būseną

5.1. Apsisaugoti nuo bot'ų (automatizuotų prisijungimų).

5.2. Galimi variantai:

Prašymas paspausti ant paveikslukų su tam tikru vaizdu;

Įvesti skaičius, girdimus garso įrašę;

6.1. Galimi variantai:

Per nepatikimus laiškus;

Per užkrėstus failus;

Per laikmenas;

Per nepatikimas svetaines;

Per bendrą tinklą.

6.2. Galimi variantai:

Failų trynimasis ar apdorojimas;

Dauginimasis kompiuteryje, taip užimant vietą;

Kompiuterio pajėgumų mažinimas;

Ir kita.

7.1. Galimi variantai:

Dėl tam tikrų dokumentų galiojimo nuo pasirašymo laiko;

Dokumento autentiškumui patvirtinti, siekiant užtikrinti, kad jis nepakeistas po pasirašymo datos.

7.2. Kaip ir ranka pasirašyto parašo.

1 programavimo dalis

```
#include
```

```
using namespace std;
```

```
// Funkcija skirta duomenų nuskaitymui ir skaičiavimams
```

```
void Skaitymas(int &n, int eile[], int &eilesInd, int rungtynes[], double santykis[], double vidurkis[]) {
```

```
    ifstream fd("U1.txt");
```

```
    int nr;
```

```
    double t[6];
```

```
    double pataike[101] = ;
```

```
    double mete[101] = ;
```

```
    fd >> n;
```

```
    for (int i = 0; i < n; i++) {
```

```
        fd >> nr >> t[0] >> t[1] >> t[2] >> t[3] >> t[4] >> t[5];
```

```
        // Patikriname, ar žaidėjas su tokiu numeriu jau buvo
```

```
        bool radom = false;
```

```
        for (int j = 0; j < eilesInd; j++) {
```

```
            if (eile[j] == nr) {
```

```
                radom = true;
```

```
                break;
```

```
            }
```

```
        }
```

```
        // Jeigu ne, pridedame jį į eilę
```

```
        if (!radom) {
```

```
            eile[eilesInd++] = nr;
```

```
        }
```

```
        rungtynes[nr]++;
```

```
        vidurkis[nr] += t[0] * 1 + t[2] * 2 + t[4] * 3;
```

```

    pataike[nr] += t[0] + t[2] + t[4];
    mete[nr] += t[1] + t[3] + t[5];
}
// Vidurkio ir metimų santykio apskaičiavimas
for (int i = 0; i < 101; i++) {
    if (rungtynes[i] > 0) {
        vidurkis[i] /= rungtynes[i];
        santykis[i] = pataike[i] / mete[i] * 100;
    }
}
fd.close();
}
int main() {
    int n, rungtynes[101] = ; // n - žaidėjų kiekis
    double vidurkis[101] = , santykis[101] = ;
    int eile[101] = ;
    int eilesInd = 0; // eilesInd - kintamasis kuriame saugojame kiek yra žaidėjų su unikaliais numeriais
    Skaitymas(n, eile, eilesInd, rungtynes, santykis, vidurkis);
    // leškom didžiausio rungtynių kiekio
    int maxRung = 0;
    for(int i = 0; i < 101; i++) {
        if(maxRung < rungtynes[i]) maxRung = rungtynes[i];
    }
    // Spausdiname rezultatus
    ofstream fr("U1rez.txt");
    fr << maxRung << endl;
    for (int i = 0; i < eilesInd; i++) {
        int nr = eile[i];
        if (rungtynes[nr] == maxRung) {
            fr << nr << " " << fixed << setprecision(1) << vidurkis[nr] << " " << setprecision(0) << " " << santykis[nr] << endl;
        }
    }
    fr.close();
    return 0;
}

```

2 programavimo dalis

```

// Antroji programavimo užduotis
#include
#include
using namespace std;
struct kambarys{ // Struktūra pradiniam duomenim apie pabėgimo kambarį išsaugoti
    string diena;
    int valandu_sk;
    int Valandos[24];
    int Tinka[24] = ; // Kintamasis naudojamas sekti kiek kiekvienos dienos kiekvieną valandą draugų galėtų
    dalyvauti
};

struct draugai{ // Struktūra pradiniam duomenim apie draugus išsaugoti
    string vardas;
    int variantu_sk;
    string Diena[168];
    int Valanda[168];
};

```

```

struct atrinkti_laikai{ // Struktūra išsauganti galutinį atrinktų valandų ir galinčių dalyvauti draugų sąrašą
    string diena;
    int valanda;
    int kiekis;
    string Vardai[20];
};

void nuskaitymas(int &d, int &dr, kambarys K[], draugai D[]); // Duomenų iš failo nuskaitymo funkcija
void skaiciavimas(int d, int dr, kambarys K[], draugai D[]); // Funkcija, apskaičiuojanti kiek mokinių tinka
kiekvienos dienos kiekviena valanda
void atrinkimas(int d, kambarys K[], int &a, atrinkti_laikai A[], int dr, draugai D[]); // Funkcija, atrenkanti tas
dienes, kai draugų buvo ne mažiau kaip keturi, ir atrenkanti jų vardus
void rikiavimas(int a, atrinkti_laikai A[]); // Funkcija rikiuojanti tinkamų laikų bei mokinių sąrašus
void isvedimas(int a, atrinkti_laikai A[]); // Duomenų išvedimo į failą funkcija
int main()
{
    int d, dr, a = 0;
    kambarys K[7];
    draugai D[20];
    atrinkti_laikai A[168];
    nuskaitymas(d, dr, K, D);
    skaiciavimas(d, dr, K, D);
    atrinkimas(d, K, a, A, dr, D);
    rikiavimas(a, A);
    isvedimas(a, A);

    return 0;
}

void nuskaitymas(int &d, int &dr, kambarys K[], draugai D[]){
    ifstream fd("U2.txt");
    char T1[5], T2[14]; // Laikini kintamieji simbolių nuskaitymui
    fd >> d >> dr;
    for(int i = 0; i < d; i++){
        fd.ignore();
        fd.get(T1, 4);
        K[i].diena = T1;
        fd >> K[i].valandu_sk;
        for (int j = 0; j < K[i].valandu_sk; j++){
            fd >> K[i].Valandos[j];
        }
    }
}

for(int i = 0; i < dr; i++){
    fd.ignore();
    fd.get(T2, 13);
    D[i].vardas = T2;
    fd >> D[i].variantu_sk;
    for (int j = 0; j < D[i].variantu_sk; j++){
        fd.ignore();
        fd.get(T1, 4);
        D[i].Diena[j] = T1;
        fd >> D[i].Valanda[j];
    }
}

fd.close();
}

void skaiciavimas(int d, int dr, kambarys K[], draugai D[]){
    for (int i = 0; i < d; i++){

```

```

for(int j = 0; j < K[i].valandu_sk; j++){
    for(int l = 0; l < dr; l++){
        for (int k = 0; k < D[l].variantu_sk; k++){
            // Jei kažkuri diena ir valanda tinka konkrečiam žmogui, tai padidiname tokių žmonių skaičių vienetu
            // prie atitinkamos dienos valandos
            if(K[i].diena == D[l].Diena[k] && K[i].Valandos[j] == D[l].Valanda[k]){
                K[i].Tinka[j]++;
            }
        }
    }
}

void atrinkimas(int d, kambarys K[], int &a, atrinkti_laikai A[], int dr, draugai D[]){
    // Atrenkame konkrečių valandų sąrašą
    for (int i = 0; i < d; i++){
        for (int j = 0; j < K[i].valandu_sk; j++){
            if (K[i].Tinka[j] >= 4){
                A[a].diena = K[i].diena;
                A[a].valanda = K[i].Valandos[j];
                A[a].kiekis = K[i].Tinka[j];
                a++;
            }
        }
    }

    int temp; // Laikinas kintamasis, skirtas vardams į struktūrą įrašyti
    for(int i = 0; i < a; i++){
        temp = 0;
        for(int j = 0; j < dr; j++){
            for(int l = 0; l < D[j].variantu_sk; l++){
                // Jei randame, kad konkrečiam žmogui tinka atrinkta diena ir valanda, įsirašome jo vardą į sąrašą
                if(A[i].diena == D[j].Diena[l] && A[i].valanda == D[j].Valanda[l]){
                    A[i].Vardai[temp] = D[j].vardas;
                    temp++;
                }
            }
        }
    }

    void rikiavimas(int a, atrinkti_laikai A[]){
        // Pirma surikiuojame atrinktų valandų sąrašą
        for(int i = 0; i < a-1; i++){
            for(int j = i+1; j < a; j++){
                if(A[i].kiekis < A[j].kiekis){
                    swap(A[i], A[j]);
                }
            }
        }

        // Tuomet surikiuojame kiekvienos valandos vardų sąrašą
        for(int i = 0; i < a; i++){
            for (int j = 0; j < A[i].kiekis - 1; j++){
                for(int l = j+1; l < A[i].kiekis; l++){
                    if(A[i].Vardai[j] > A[i].Vardai[l]){
                        swap(A[i].Vardai[j], A[i].Vardai[l]);
                    }
                }
            }
        }
    }
}

```

```
}  
}  
void isvedimas(int a, atrinkti_laikai A){  
    ofstream fr("U2rez.txt");  
    for(int i = 0; i < a; i++){  
        fr << A[i].diena << " " << A[i].valanda << " " << A[i].kiekis << endl;  
        for(int j = 0; j < A[i].kiekis; j++){  
            fr << A[i].Vardai[j] << endl;  
        }  
    }  
    fr.close();  
}
```