

Pagrindiniai algoritmai dirbant su sveikųjų ir realiųjų skaičių masyvų reikšmėmis

Sumos skaičiavimo algoritmas

Sveikieji skaičiai	Realieji skaičiai
<pre>int Suma (int X[], int n){ int s = 0; for (int i = 0; i < n; i++) s = s + X[i]; return s; }</pre>	<pre>double Suma (double X[], int n){ double s = 0; for (int i = 0; i < n; i++) s = s + X[i]; return s; }</pre>

Kiekio skaičiavimo algoritmas (grąžina, kiek teigiamų reikšmių yra masyve)

Sveikieji skaičiai	Realieji skaičiai
<pre>int Kiekis (int X[], int n){ int k = 0; for (int i = 0; i < n; i++) if (X[i] > 0) k++; return k; }</pre>	<pre>int Kiekis (double X[], int n){ int k = 0; for (int i = 0; i < n; i++) if (X[i] > 0) k++; return k; }</pre>

Didžiausios masyvo reikšmės vieta masyve

Sveikieji skaičiai	Realieji skaičiai
<pre>int Didziausia (int X[], int n){ int m = 0; for (int i = 0; i < n; i++) if (X[i] > X[m]) m = i; return m; }</pre>	<pre>int Didziausia (double X[], int n){ int m = 0; for (int i = 0; i < n; i++) if (X[i] > X[m]) m = i; return m; }</pre>

Mžiausios masyvo reikšmės vieta masyve

Sveikieji skaičiai	Realieji skaičiai
<pre>int Maziausia (int X[], int n){ int m = 0; for (int i = 0; i < n; i++) if (X[i] < X[m]) m = i; return m; }</pre>	<pre>int Maziausia (double X[], int n){ int m = 0; for (int i = 0; i < n; i++) if (X[i] < X[m]) m = i; return m; }</pre>

Reikšmės šalinimo algoritmas

Sveikieji skaičiai	Realieji skaičiai
<pre>void Salinimas (int X[], int & n, int k) { for (int i = k; i < n; i++) X[i] = X[i + 1]; n--; }</pre>	<pre>void Salinimas (double X[], int & n, int k) { for (int i = k; i < n; i++) X[i] = X[i + 1]; n--; }</pre>

Reikšmės įterpimo algoritmas

Sveikieji skaičiai	Realieji skaičiai
<pre>void Iterpimas (int X[], int & n, int k, int koks) { for (int i = n; i > k; i--) X[i] = X[i - 1]; n++; X[k] = koks; }</pre>	<pre>void Iterpimas (double X[], int & n, int k, double koks) { for (int i = n; i > k; i--) X[i] = X[i - 1]; n++; X[k] = koks; }</pre>

Rikiavimo algoritmas

Sveikieji skaičiai	Realieji skaičiai
<pre>void Rikiavimas(int X[], int n){ int m; int sukeitimas; for (int i = 0; i < n - 1; i++) { m = i; for (int j = i + 1; j < n; j++) if (X[j] > X[m]) m = j; sukeitimas = X[i]; X[i] = X[m]; X[m] = sukeitimas; } }</pre>	<pre>void Rikiavimas(double X[], int n){ int m; double sukeitimas; for (int i = 0; i < n - 1; i++) { m = i; for (int j = i + 1; j < n; j++) if (X[j] > X[m]) m = j; sukeitimas = X[i]; X[i] = X[m]; X[m] = sukeitimas; } }</pre>

Struktūrų masyvai. Pagrindiniai algoritmai

Struktūra:

```
struct Mokinys {  
    string vardas;  
    int amzius;  
    int ugis;  
    int pinigai;  
};
```

1. Sumos skaičiavimas: kiek pinigų turi visi mokiniai

```
int Suma (Mokinys M[], int n) {  
    int s = 0;  
    for (int i = 0; i < n; i++)  
        s = s + M[i].pinigai;  
    return s;  
}
```

2. Vidurkio skaičiavimas: koks vidutinis mokinio ūgis

```
double Vidurkis (Mokinys M[], int n) {  
    double v = 0;  
    for (int i = 0; i < n; i++)  
        v = v + M[i].ugis;  
    return (double) v / n;  
}
```

3. Kiekio skaičiavimas: kiek yra mokinių, kurių amžius ne didesnis už 15 metų:

```
int Keli (Mokinys M[], int n) {  
    int k = 0;  
    for (int i = 0; i < n; i++)  
        if (M[i].amzius <= 15) k++;  
    return k;  
}
```

4. Mažiausios reikšmės vietos paieška: kuris mokinys yra jauniausias

Isidėmėkite: gražinamas masyvo indeksas!!!!

```
int Jauniausias (Mokinys M[], int n) {  
    int m = 0;  
    for (int i = 0; i < n; i++)  
        if (M[m].amzius > M[i].amzius) m = i;  
    return m;  
}
```

5. Didžiausios reikšmės vietos paieška: kuris mokinys yra aukščiausias
Isidėmėkite: gražinamas masyvo indeksas!!!!

```
int Auksciausias (Mokinys M[], int n) {
    int m = 0;
    for (int i = 0; i < n; i++)
        if (M[m].ugis > M[i].ugis) m = i;
    return m;
}
```

6. Elemento šalinimas iš struktūrų masyvo: k-tojo mokinio šalinimas

```
void Salinimas (Mokinys M[], int & n, int k) {
    for (int i = k; i < n; i++)
        M[i] = M[i + 1];
    n--;
}
```

7. Elemento įterpimas į struktūrų masyvą: į k-tąją vietą įterpti naują mokinį *naujas*

```
void Ierpimas (Mokinys M[], int & n, int k, Mokinys naujas) {
    for (int i = n; i > k; i--)
        M[i] = M[i - 1];
    n++;
    M[k] = naujas;
}
```

8. Struktūrų masyvo elementų rikiavimas: surikiuoti mokinius pagal ūgį didėjimo kryptimi, jei ūgiai sutampa, tuomet pagal abėcėlę nuo A iki Z

```
void Rikiavimas(Mokinys M[], int n){
    int m; Mokinys sukeitimas;
    for (int i = 0; i < n - 1; i++) {
        m = i;
        for (int j = i + 1; j < n; j++)
            if ((X[j].ugis > X[m].ugis) ||
                ((X[j].ugis == X[m].ugis) &&
                 (X[j].vardas < X[m].vardas))) m = j;
        sukeitimas = X[i]; X[i] = X[m]; X[m] = sukeitimas;
    }
}
```

arba naudojant **swap**:

```
void Rikiavimas(Mokinys M[], int n){
    int m;
    for (int i = 0; i < n - 1; i++) {
        m = i;
        for (int j = i + 1; j < n; j++)
            if ((X[j].ugis > X[m].ugis) ||
                ((X[j].ugis == X[m].ugis) &&
                 (X[j].vardas < X[m].vardas))) m = j;
        swap(M[i], M[m]);
    }
}
```