

# Programozás

## (GKxB\_INTM114)

Dr. Hatwagner F. Miklós

Széchenyi István Egyetem, Győr

[https://github.com/wajzy/GKxB\\_INTM114.git](https://github.com/wajzy/GKxB_INTM114.git)

2024. február 24.

## haromszog1.cpp Háromszög szerkeszthetősége; az oldalhossz beolvasása 3x ismétlődik!

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int a, b, c;
5      bool megszerkesztheto = false;
6      cout << "Adja meg egy haromszog oldalhosszait!\n";
7      do {
8          // hatultesztelo ciklus eleje...
9          cout << "A oldal hossza: "; cin >> a;
10         } while(a <= 0); // ...es vege
11         do {
12             cout << "B oldal hossza: "; cin >> b;
13         } while(b <= 0);
14         do {
15             cout << "C oldal hossza: "; cin >> c;
16         } while(c <= 0);
17         if(a+b<=c or b+c<=a or c+a<=b) {
18             cout << "Ez nem szerkesztheto meg!\n";
19         } else {
20             megszerkesztheto = true;
21             cout << "Megszerkesztheto.\n";
22         }
23     } while(not megszerkesztheto);
24     return 0; }
```

## haromszog3.cpp

```
1  #include <iostream>
2  #define OLDALSZAM 3
3  using namespace std;
4  int main() {
5      int ot[OLDALSZAM], i;
6      bool megszerkesztheto = false;
7      char onev;           // aktualis oldal megnevezese
8      cout << "Adja meg egy haromszog oldalhosszait!\n";
9      do {
10         i = 0; onev = 'A';
11         while(i < OLDALSZAM) {
12             do {
13                 cout << onev << " oldal hossza: "; /* oldal megnevezese */ cin >> ot[i];
14             } while(ot[i] <= 0);
15             i++; onev++; // kovetkezo oldal neve
16         }
17         if (ot[0]+ot[1]<=ot[2] or ot[1]+ot[2]<=ot[0] or ot[2]+ot[0]<=ot[1]) {
18             cout << "Ez nem szerkesztheto meg!\n";
19         } else {
20             megszerkesztheto = true;
21             cout << "Megszerkesztheto.\n";
22         }
23     } while(not megszerkesztheto);
24     return 0; }
```

## Tömb definíció

- *típus tömbazonosító[méret];*
- `pl. int ot[3];`
- *méret* pozitív, egész értékű *állandó kifejezés*
- *állandó kifejezés* értéke fordítási időben kiszámítható

## Tömb tárigénye

$$\text{sizeof}(\text{tömbazonosító}) \equiv \text{méret} * \text{sizeof}(\text{típus})$$

## Tömbelemek (indexes változó) elérése

- *tömbazonosító[index]*
- $0 \leq \text{index} \leq \text{méret} - 1$

## Oldal nevének előállítás

- Kihasználjuk, hogy az **ASCII kódok** a betűk abc-beli sorrendjének megfelelően növekednek ('A' == 65, 'B' == 66, ..., 'Z' == 90)
- Hasonló a helyzet a számjegyekkel is ('0' == 48, '1' == 49, ..., '9' == 57)
- Számjegy → ASCII kód: '0' + számjegy
- ASCII kód → Számjegy: karakter - '0'
- Betűk is hasonlóan kezelhetők

## szamlalo1.cpp Számjegy karakterek számlálása, 1/2

```
1  #include <iostream>
2  #include <cstdio>
3  using namespace std;
4  int main(void) {
5      cout << "Számjegyek, ures- es egyeb karakterek leszamlalasa\n"
6           << "a bemeneten EOF-ig vagy Ctrl+D-ig.\n\n";
7      int k, feher=0, egyeb=0;
8      int nulla=0, egy=0, ketto=0, harom=0, negy=0, ot=0, hat=0, het=0, nyolc=0, kilenc=0; // :(
9      while((k=cin.get()) != EOF){
10         switch(k) { // Ronda, mint egy loharapas!
11             case '0': nulla++; break;
12             case '1': egy++; break;
13             case '2': ketto++; break;
14             case '3': harom++; break;
15             case '4': negy++; break;
16             case '5': ot++; break;
17             case '6': hat++; break;
18             case '7': het++; break;
19             case '8': nyolc++; break;
20             case '9': kilenc++; break;
21             case '\n': case '\t': feher++; break;
22             default: egyeb++; break;
23         }
24     }
```

## szamlalo1.cpp 2/2

```
25  cout << "Szamjegyek:\n";
26  cout << "Nulla:\t" << nulla << " db\n"; // Uram irgalmazz!
27  cout << "Egy:\t" << egy << " db\n";
28  cout << "Ketto:\t" << ketto << " db\n";
29  cout << "Harom:\t" << harom << " db\n";
30  cout << "Negy:\t" << negy << " db\n";
31  cout << "Ot:\t" << ot << " db\n";
32  cout << "Hat:\t" << hat << " db\n";
33  cout << "Het:\t" << het << " db\n";
34  cout << "Nyolc:\t" << nyolc << " db\n";
35  cout << "Kilenc:\t" << kilenc << " db\n";
36  cout << "Ures karakterek: " << feher << ", egyeb: " << egyeb << endl;
37  return 0;
38 }
```

Nyilvánvalóan szükségünk van egy tömbre!

## szamlalo2.cpp 1/2

```
1 #include <iostream>
2 #include <cstdio>
3 #define DB 10
4 using namespace std;
5
6 int main(void) {
7     cout << "Szamjegyek, ures- es egyeb karakterek leszamlalasa\n"
8         << "a bemeneten EOF-ig vagy Ctrl+D-ig.\n\n";
9     int i, k, feher=0, egyeb=0;
10    int szamjegy[DB]; // 10 elemu tomb a tiz szamjegynek
11    i = 0;
12    while(i < DB) {
13        szamjegy[i] = 0; // Szamlalok nullazasa
14        i++;
15    }
```



## szamlalo2.cpp 2/2

```
16 while((k=cin.get()) != EOF){
17     if(k>='0' and k<='9') {
18         i = k - '0';        // Karakter atalakítása számmá,
19         szamjegy[i]++;      // amit indexként használunk
20     } else if(k==' ' or k=='\n' or k=='\t') fehér++;
21     else egyeb++;
22 }
23 cout << "Szamjegyek:\n";
24 i = 0;                        // Eredmenyek kijelzese
25 while(i < DB) {
26     cout << i << '\t' << szamjegy[i] << " db\n";
27     i++;
28 }
29 cout << "Ures karakterek: " << fehér << ", egyeb: " << egyeb << endl;
30 return 0;
31 }
```

## Tömbelemek, mint számlálók

Az *i* számjegy darabszámát **szamjegy[i]** tárolja! (Azaz pl. 0-ból szamjegy[0], 1-ből szamjegy[1], stb. érkezett.)

## Tömbök inicializálása

- *típus tömbazonosító[<méret>]<=inicializátorlista>;*
- Ha *inicializátorlista* elemszáma  $< méret \rightarrow$  további elemek nullázódnak
- Ha *inicializátorlista* elemszáma  $> méret \rightarrow$  hiba!
- Ha *méret*-et nem specifikáltak, a fordító megállapítja *inicializátorlista* elemszámából
- De a *méret* és az *inicializátorlista* közül legalább az egyiknek léteznie kell!

## szamlalo3.cpp

```
1 #include <iostream>
2 #include <cstdio>
3 #define DB 10
4 using namespace std;
5
6 int main(void) {
7     cout << "Szamjegyek, ures- es egyeb karakterek leszamlalasa\n"
8         << "a bemeneten EOF-ig vagy Ctrl+D-ig.\n\n";
9     int k, i, feher=0, egyeb=0;
10    int szamjegy[DB] = {0}; // szamlalok nullazasa inicializalassal
11    while((k=cin.get()) != EOF){
12        if(k>='0' && k<='9') ++szamjegy[k-'0']; // szamlalok leptetese
13        else if(k==' ' or k=='\n' or k=='\t') ++feher;
14        else ++egyeb;
15    }
16    cout << "Szamjegyek:\n";
17    i = 0;
18    while(i < DB) {
19        cout << i << '\t' << szamjegy[i] << " db\n"; i++;
20    }
21    cout << "Ures karakterek: " << feher << ", egyeb: " << egyeb << endl;
22    return 0;
23 }
```

## forditva1.cpp

```
1  #include <iostream>
2  #define N 5
3  using namespace std;
4
5  int main() {
6      cout << "Adjon meg " << N << " számot, kiírjuk őket fordított sorrendben!\n\n";
7      int szamok[N], db=0;
8      while(db < N) {
9          cout << db+1 << ". szám: ";
10         cin >> szamok[db++];
11     }
12     cout << "\nFordított sorrendben:\n";
13     while(db-->0) {
14         cout << szamok[db] << '\t';
15     }
16     cout << endl;
17     return 0;
18 }
```

## binker1.cpp Bináris keresés, 1/4

```
1 #include <iostream>
2 #define N 10
3 using namespace std;
4 // Bináris keresés csak növekvőleg rendezett elemekkel használható!
5 int main() { // 0 1 2 3 4 5 6 7 8 9
6     int szamok[N] = {-23, -11, 0, 1, 7, 13, 14, 17, 21, 42};
7     cout << "Mit keresünk? ";
8     int szam;
9     cin >> szam; // szam == 1
10    int also=0, felso=N-1, kozep; // also == 0, felso == 9
11    while(also <= felso) {
12        kozep = (also+felso)/2; // kozep == 4
13        if(szam < szamok[kozep]) felso = kozep-1; // felso == 3
14        else if(szam > szamok[kozep]) also = kozep+1;
15        else {
16            cout << "Megtaláltuk a(z) " << kozep << " indexu helyen.\n";
17            return 0;
18        }
19    }
20    cout << "Nem találtuk meg, de a(z) " << also << " indexu elemben "
21        << "lenne a helye.\n";
22    return 0;
23 }
```

## binker2.cpp Bináris keresés, 2/4

```
1  #include <iostream>
2  #define N 10
3  using namespace std;
4  // Bináris keresés csak növekvőleg rendezett elemekkel használható!
5  int main() {           // 0   1   2   3   4   5   6   7   8   9
6      int szamok[N] = {-23, -11, 0, 1, 7, 13, 14, 17, 21, 42};
7      cout << "Mit keresünk? ";
8      int szam;
9      cin >> szam;                // szam == 1
10     int also=0, felso=N-1, kozep;
11     while(also<=felso) {         // also == 0, felso == 3
12         kozep = (also+felso)/2;   // kozep == 1
13         if(szam < szamok[kozep]) felso = kozep-1;
14         else if(szam > szamok[kozep]) also = kozep+1; // also == 2
15         else {
16             cout << "Megtaláltuk a(z) " << kozep << " indexu helyen.\n";
17             return 0;
18         }
19     }
20     cout << "Nem találtuk meg, de a(z) " << also << " indexu elemben "
21         << "lenne a helye.\n";
22     return 0;
23 }
```

## binker3.cpp Bináris keresés, 3/4

```
1 #include <iostream>
2 #define N 10
3 using namespace std;
4 // Bináris keresés csak növekvőleg rendezett elemekkel használható!
5 int main() { // 0 1 2 3 4 5 6 7 8 9
6     int szamok[N] = {-23, -11, 0, 1, 7, 13, 14, 17, 21, 42};
7     cout << "Mit keresünk? ";
8     int szam;
9     cin >> szam; // szam == 1
10    int also=0, felso=N-1, kozep;
11    while(also<=felso) { // also == 2, felso == 3
12        kozep = (also+felso)/2; // kozep == 2
13        if(szam < szamok[kozep]) felso = kozep-1;
14        else if(szam > szamok[kozep]) also = kozep+1; // also == 3
15        else {
16            cout << "Megtalaltuk a(z) " << kozep << " indexu helyen.\n";
17            return 0;
18        }
19    }
20    cout << "Nem talaltuk meg, de a(z) " << also << " indexu elemben "
21        << "lenne a helye.\n";
22    return 0;
23 }
```

## binker4.cpp Bináris keresés, 4/4

```
1 #include <iostream>
2 #define N 10
3 using namespace std;
4 // Bináris keresés csak növekvőleg rendezett elemekkel használható!
5 int main() { // 0 1 2 3 4 5 6 7 8 9
6     int szamok[N] = {-23, -11, 0, 1, 7, 13, 14, 17, 21, 42};
7     cout << "Mit keresünk? ";
8     int szam;
9     cin >> szam; // szam == 1
10    int also=0, felso=N-1, kozep;
11    while(also<=felso) { // also == 3, felso == 3
12        kozep = (also+felso)/2; // kozep == 3
13        if(szam < szamok[kozep]) felso = kozep-1;
14        else if(szam > szamok[kozep]) also = kozep+1;
15        else { // Megtaláltuk!
16            cout << "Megtaláltuk a(z) " << kozep << " indexu helyen.\n";
17            return 0;
18        }
19    }
20    cout << "Nem találtuk meg, de a(z) " << also << " indexu elemben "
21        << "lenne a helye.\n";
22    return 0;
23 }
```



## buborek.cpp Buborékrendezés, elemcsere segédváltozó nélkül

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int szamok[] = {12, 3, 54, -4, 56, 4, 7, 3};
6      int n = sizeof(szamok)/sizeof(szamok[0]); // Tomb elemszam szamolasa
7      int i=n-1, k;
8      while(i>=1) {
9          k = 0;
10         while(k<i) {
11             if(szamok[k]>szamok[k+1]) {
12                 int csere = szamok[k]; szamok[k] = szamok[k+1]; szamok[k+1] = csere;
13             }
14             k++;
15         }
16         i--;
17     }
18     cout << "Rendezes utan:\n"; i = 0;
19     while(i < n) {
20         cout << szamok[i] << '\t'; i++;
21     }
22     cout << endl;
23     return 0;
24 }
```

## string.cpp string demo

```
1 #include <iostream>
2 using namespace std;
3
4 int main(void) {
5     string s1;
6     string s2 = " es Bolka";           // inicializalas
7     cout << "s1: " << s1 << '\n';    // ures string!
8     s1 = "Lolka";                     // literal ertekul adható stringnek
9     s1 += s2;                          // stringek összefuzhetőek
10    cout << s1 << "\nAdjon meg egy szot! ";
11    cin >> s2;                          // Szó beolvasása stringbe
12    cout << "A szó hossza: " << s2.length() << '\n';
13    // stringek összehasonlíthatóak relációs operátorokkal
14    if(s2 == "Frakk") cout << "Megint a régi mese\n";
15    cout << s2 << " első betűje: " << s2[0] << endl;
16    return 0;
17 }
```

## kettes1.cpp 2 → 10

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      string b;
6      unsigned d, i;
7      cout << "Adjon meg egy kettes szamrendszerbeli szamot!\n";
8      cin >> b;
9      d = i = 0;
10     while(i < b.length()) {
11         d = d*2 + b[i] - '0'; i++;
12     }
13     cout << "Tizes szamrendszerben: " << d << endl;
14     return 0;
15 }
```

## kettes2.cpp 10 → 2

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      string b; char c; int d;
5      cout << "Adjon meg egy tízes számrendszerbeli számot!\n";
6      cin >> d;
7      while(d > 0) {
8          c = d%2 + '0'; b = c + b; d /= 2;
9      }
10     cout << "Kettes számrendszerben: " << b << endl;
11     return 0;
12 }
```

## neptun1.cpp

```
1  #include <iostream>
2  using namespace std;
3  int main(void) {
4      bool helytelen;
5      string neptun;          // karakterlanc tarolasara szolgalo C++ tipus
6      do {
7          helytelen = false;
8          cout << "Adja meg a Neptun kodjat: "; cin >> neptun;
9          if(neptun.length() != 6) { // karakterlanc hosszának lekerdezese
10             cout << "Hat karakterbol kell allnia!\n"; helytelen = true;
11         } else {
12             unsigned i=0;
13             while(not helytelen and i<neptun.length()) {
14                 char k = neptun[i];
15                 bool szamjegy = k>='0' and k<='9';
16                 bool nagybetu = k>='A' and k<='Z';
17                 bool kisbetu = k>='a' and k<='z';
18                 if(not szamjegy and not nagybetu and not kisbetu) {
19                     cout << "Csak szamjegyeket es betuket tartalmazhat!\n";
20                     helytelen = true; }
21                 i++; } }
22     } while(helytelen);
23     cout << "Rendben.\n";
24     return 0; }
```

## neptun2.cpp

```
1  #include <iostream>
2  #include <cctype> // toupper() miatt
3  using namespace std;
4  int main(void) {
5      bool helytelen;
6      string neptun;
7      do {
8          helytelen = false;
9          cout << "Adja meg a Neptun kodjat: "; cin >> neptun;
10         if(neptun.length() != 6) { // karakterlanc hosszának lekerdezese
11             cout << "Hat karakterbol kell allnia!\n"; helytelen = true;
12         } else {
13             unsigned i=0;
14             while(not helytelen and i<neptun.length()) {
15                 char k = toupper(neptun[i]); // nagybeture alakitas
16                 if((k<'0' or k>'9') and (k<'A' or k>'Z')) {
17                     cout << "Csak szamjegyeket es betuket tartalmazhat!\n";
18                     helytelen = true; }
19                 i++; } }
20     } while(helytelen);
21     cout << "Rendben.\n";
22     return 0; }
```

## Karakterek osztályozása, átalakítása

- `cctype` vagy `ctype.h` beszerkesztése szükséges
- Függvények vagy makrók (előfeldolgozó)
- Paraméter típusa `int`, de az értéknek `unsigned char`-ral ábrázolhatónak, vagy EOF-nak kell lennie
- Visszatérési érték `int`, karakterosztályozó rutinoknál logikai értéként kezelendő

Fv./makró név	Funkció	Fv./makró név	Funkció
<code>islower(c)</code>	c kisbetű?	<code>isxdigit(c)</code>	c hexadecimális számjegy?
<code>isupper(c)</code>	c nagybetű?	<code>isspace(c)</code>	c fehér karakter?
<code>isalpha(c)</code>	c betű?	<code>isprint(c)</code>	c nyomtatható?
<code>isdigit(c)</code>	c számjegy?	<code>tolower(c)</code>	c kisbetűs alakja, ha c nagybetű
<code>isalnum(c)</code>	c alfanumerikus?	<code>toupper(c)</code>	c nagybetűs alakja, ha c kisbetű

## neptun3.cpp

```
1  #include <iostream>
2  #include <cctype> // isalnum() miatt
3  using namespace std;
4  int main(void) {
5      bool helytelen;
6      string neptun;
7      do {
8          helytelen = false;
9          cout << "Adja meg a Neptun kodjat: "; cin >> neptun;
10         if(neptun.length() != 6) { // karakterlanc hosszának lekerdezese
11             cout << "Hat karakterbol kell allnia!\n"; helytelen = true;
12         } else {
13             unsigned i=0;
14             while(not helytelen and i<neptun.length()) {
15                 if(not isalnum(neptun[i])) { // alfanumerikus karakter?
16                     cout << "Csak szamjegyeket es betuket tartalmazhat!\n";
17                     helytelen = true; }
18                 i++; } }
19     } while(helytelen);
20     cout << "Rendben.\n";
21     return 0; }
```