

# 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. március 25.

## zh1.cpp Legeredményesebb hallgatók kikeresése

```

1 #include <iostream>
2 #define MAXLETSZAM 1000
3 using namespace std;
4
5 int main(void) {
6     string nevek[MAXLETSZAM];
7     int pontszamok[MAXLETSZAM];
8     int letszam, maxPont = 0;
9     cout << "Legeredmenyesebb hallgatók kikeresése\n"
10         << "Osztálylétszám: ";
11     cin >> letszam;
12     for(int i=0; i<letszam; i++) {
13         cout << (i+1) << ". hallgató neve: "; cin >> nevek[i];
14         cout << "ZH-n elért pontszám: "; cin >> pontszamok[i];
15         if(pontszamok[i] > maxPont) maxPont = pontszamok[i];
16     }

```

## zh1.cpp

```
17     cout << "Legmagasabb pontszám: " << maxPont
18         << ", akik elertek:\n";
19     for(int i=0; i<letszam; i++) {
20         if(pontszamok[i] == maxPont) {
21             cout << nevek[i] << '\n';
22         }
23     }
24     return 0;
25 }
```

### Probléma:

- egy hallgató neve és pontszáma szorosabb logikai kapcsolatban van, mint pl. a különféle hallgatók nevei
- a tömbök viszont ezt nem tükrözik

- logikailag összetartozó változók csoportjának egyszerűbb kezelése
- összetett, felhasználói adattípus hozható létre
- egy vagy több, elnevezett *tag* együttese, önálló azonosítóval
- lehetőségek:
  - hozzárendelés (másolás)
  - átadhatók függvénynek paraméterként
  - lehet függvény visszatérési értéke
- **nem lehetséges: összehasonlítás** (esetleg tagonként)
- struktúrtag *szinte* bármiből lehet

## zh2.cpp Legeredményesebb hallgatók kikeresése

```
1  #include <iostream>
2  #define MAXLETSZAM 1000
3  using namespace std;
4
5  struct hallgato {
6      string nev;
7      int pontszam;
8  };
9
10 int main(void) {
11     hallgato hg[MAXLETSZAM];
12     int letszam, maxPont = 0;
13     cout << "Legeredményesebb hallgatók kikeresése\n"
14           << "Osztálylétszám: ";
15     cin >> letszam;
```

## zh2.cpp

```
16  for(int i=0; i<letszam; i++) {
17      cout << (i+1) << ". hallgató neve: "; cin >> hg[i].nev;
18      cout << "ZH-n elert pontszam: "; cin >> hg[i].pontszam;
19      if(hg[i].pontszam > maxPont) maxPont = hg[i].pontszam;
20  }
21  cout << "Legmagasabb pontszam: " << maxPont << ", akik elertek:\n";
22  for(int i=0; i<letszam; i++) {
23      if(hg[i].pontszam == maxPont) {
24          cout << hg[i].nev << '\n';
25      }
26  }
27  return 0;
28 }
```

Általános alak: **struct** <struktúracímke> <struktúratag-deklarációlista> <azonosítólista>;

### Struktúradeklaráció példa

```
struct hallgato { // Struktúra deklarálása
    string nev;
    int pontszam;
};

struct hallgato mari; // Változók definiálása
hallgato karcsi, hg[1000];
```

- hallgato a struktúra címkéje, a típust azonosítja:  
struct hallgato mari;  
hallgato karcsi;
- Tagok: nev, pontszam (egyedi azonosítók)
- Változók: mari, karcsi  
hg[1000] egy 1000 elemű struktúratömb

## Hol deklaráljuk a struktúrát?

- a típus első felhasználása előtt
- jellemzően a forrás elején, minden függvényen kívül

Minden deklaráció *egyedi típust hoz létre*, melyek nem azonosak akkor sem, ha szerkezetük megegyezik.

### Típusok különbözősége

```
struct hallgato1 {  
    string nev; int pontszam;  
};  
  
struct hallgato2 {  
    string nev; int pontszam;  
};  
  
hallgato1 mari;  
hallgato2 gizi;  
gizi = mari; // error: no match for 'operator='  
              // (operand types are 'main()::hallgato2' and 'main()::hallgato1')
```



- Struktúratag lehet pl.
  - korábban definiált szerkezetű struktúra
  - beágyazott struktúra, akár címke nélkül is
  - tömb
  - (függvény → következő félév anyaga)
- A tag azonosítójának csak a struktúrán belül kell egyedinek lennie
- A deklaráció végén lévő pontosvessző nem hagyható el!

### Helyes tagdeklarációk

```
struct s { int i; };  
struct tag_dekl {  
    struct s s1;  
    struct { int i; long l; } b;  
    int szamok[30];  
};
```

Struktúratag típusa nem lehet pl.

- void
- saját maga

### Hibás tagdeklarációk

```
struct nem_teljes;
struct tag_hiba {
    void v; /* error: variable or field 'v' declared void */
    struct nem_teljes s; /* error: field 's' has incomplete type */
    struct tag_hiba th; /* error: field 'th' has incomplete type */
};
```

Tagelérés operátor / szelekciós operátor / tagszelektor

- *struktúra.tag*
- Magas precedenciájú operátor, balról jobbra köt

### Struktúratagok elérése, értékadások

```
struct hallgato {  
    string nev;  
    string telSzamok[2];  
    struct {  
        int ev, ho, nap;  
    } szulDatum;  
};  
/* ... */  
hallgato gizi;  
gizi.nev = "Kovács Gizella";  
gizi.telSzamok[0] = "+36 1 123-4567"; gizi.telSzamok[1] = "96/123-456";  
gizi.szulDatum.ev = 1990; gizi.szulDatum.ho = 1; gizi.szulDatum.nap = 2;
```

Az inicializáció során a tagok a deklarációbeli sorrendjükben veszik fel az inicializátorlista elemeinek értékét.

Az inicializátor azonos típusú struktúra is lehet.

## Struktúra inicializálása

```
struct hallgato {  
    string nev, neptun;  
    int ev, ho, nap;  
};
```

```
hallgato gizi = { "Kovács Gizella", "A1B2C3", 1990, 4, 23 };  
hallgato mari = gizi;  
// mari = { "Nagy Maria", "ABC123", 1995, 5, 6 };  
// error: 'mari' does not name a type
```

## Struktúrába ágyazott tagok inicializálása: beágyazott inicializátorokkal

### Beágyazott struktúra és tömb inicializálása

```
struct datum {
    int ev, ho, nap;
};
struct hallgato {
    string nev, neptun;
    string telSzamok[2];
    datum szulDatum, diplomaSzerzes;
};
hallgato gizi = { "Kovács Gizella", "A1B2C3",
    {"+36 1 123-4567", "+36 20 987-6543"},
    {1990, 4, 23}, {2015, 6, 3} };
```

- Inicializátorlista elemszáma nem haladhatja meg a tagok számát!
- Ha viszont kevesebb elemű → nullázás
- Aggregátumok esetén a { } elhagyhatók, ill. valamennyi inicializátor köré is helyezhető, de *célszerű* követni az aggregátum szerkezetét

Kijelölt inicializátorok ([designated initializers](#), C++20) használata: közvetlen hivatkozás a tagokra

## Dátum struktúra inicializálása

```
struct datum {  
    int ev = { 2023 };  
    int ho = { 01 };  
    int nap = { 01 };  
} szulDatum = { .ho = 1, .nap = 26 };
```

- Csak aggregátumok inicializálására használhatóak
- (Csak nem statikus tagok inicializálhatók)
- Az inicializálók sorrendjének egyeznie kell a tagok sorrendjével
- Nem kötelező az összes tagot inicializálni
- Nem keverhető a hagyományos inicializáció a kijelölt inicializálókkal
- Egy tag egy inicializálóval inicializálható
- Nem lehet az inicializátorokat egymásba ágyazni

## naptar1.cpp Dátumkezelő függvények

```
5  struct datum {
6      int ev, ho, nap;
7  };
8
9  bool szoko(int ev) { // szokoev megallapitas
10     return (ev%4==0 and ev%100!=0) or ev%400==0;
11 }
12
13 int napok(int ev, int ho) { // honap napjainak szamat
14     int nt[12] = // adja vissza adott evben
15     { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
16     if(ho == 2) {
17         if(szoko(ev)) return 29; else return 28;
18     } else {
19         return nt[ho-1];
20     }
21 }
```

## naptar1.cpp

```
23 bool ellenoriz(datum d) { // datum tartalmi ellenorzese
24     if(d.ho<1 or d.ho>12) return false;
25     int n = napok(d.ev, d.ho);
26     if(d.nap<1 or d.nap>n) return false;
27     return true;
28 }
29
30 int evNapja(datum d) { // ev napjanak meghatarozasa
31     int n = d.nap;      // ev, ho, napbol
32     for(int h=1; h<d.ho; h++) {
33         n += napok(d.ev, h);
34     }
35     return n;
36 }
```



## naptar1.cpp

```
38 string hetNapja(datum d) { // het napjának számítása
39     int hn;
40     char seged[12] = { 6, 2, 2, 5, 0, 3, 5, 1, 4, 6, 2, 4 };
41     string napNev[7] = { "hetfo", "kedd", "szerda",
42         "csutortok", "pentek", "szombat", "vasarnap" };
43     hn = d.ev*1.25 + d.nap;
44     hn += seged[d.ho-1];
45     if ((d.ev%4==0) and d.ho<3) hn--;
46     while (hn > 7) hn -= 7;
47     return napNev[hn==1 ? 6 : hn-2];
48 }
```

## naptar1.cpp

```
50  int bazis(datum d, int bazisEv) { // bazisEv.01.01 ota eltelt napok szama
51      int b = 0;
52      for(int e=bazisEv; e<d.ev; e++) {
53          b += 365 + szoko(e);
54      }
55      for(int h=1; h<d.ho; h++) {
56          b += napok(d.ev, h);
57      }
58      b += d.nap;
59      return b;
60  }
61
62  int kulonbseg(datum tol, datum ig) {
63      int bazisEv = tol.ev < ig.ev ? tol.ev : ig.ev;
64      return bazis(ig, bazisEv) - bazis(tol, bazisEv);
65  }
```

## naptar1.cpp

```
67 datum hoEsNap(int ev, int evNapja) { // nap even beluli
68     datum d = { ev, 0, evNapja };    // szamabol ho es
69     int h, n;                        // nap szamolasa
70     for(h=1; d.nap>(n=napok(ev, h)); h++) {
71         d.nap -= n;
72     }
73     d.ho = h;
74     return d;
75 }
```

## naptar1.cpp

```
77 int main(void) {
78     datum d = {2024, 3, 20};
79     cout << "A megadott datum " << (ellenoriz(d)? "helyes": "hibas")
80         << ".\n" << d.ev << ' ' << setw(2) << setfill('0') << d.ho
81         << ' ' << setw(2) << setfill('0') << d.nap << " az ev "
82         << evNapja(d) << ". napja, " << hetNapja(d) << ".\n";
83     datum kar = {2024, 12, 24};
84     cout << "Hany nap van karacsonyig? " << kulonbseg(d, kar);
85     int evNapja = 300;
86     d = hoEsNap(d.ev, evNapja);
87     cout << '\n' << d.ev << ' ' << evNapja << ". napja: "
88         << setw(2) << setfill('0') << d.ho << ' ' << d.nap << endl;
89     return 0; }
```

## Kimenet

A megadott datum helyes.

2024.03.20 az év 80. napja, szerda.

Hány nap van karácsonyig? 279

2024 300. napja: 10.26

Az iomanip fejfájl néhány hasznos függvénye:

`setw()` Mezőszélesség állítása

`setfill()` Kitöltő karakter beállítása

`fixed` Rögzített számú tizedesjegy megjelenítése

`setprecision()` Tizedesjegyek számának beállítása

## Kimenet (1/2) – Téglalapok rajzolása

Rajzprogram - adj meg a téglalapok adatait!

1. téglalap BF sarok X: [0, 78] (negativra vege) 1

1. téglalap BF sarok Y[0, 23] 1

1. téglalap JA sarok X[2, 79] 11

1. téglalap JA sarok Y[2, 24] 11

1. téglalap rajzoló karaktere: |

2. téglalap BF sarok X: [0, 78] (negativra vege) 6

2. téglalap BF sarok Y[0, 23] 6

2. téglalap JA sarok X[7, 79] 16

2. téglalap JA sarok Y[7, 24] 16

2. téglalap rajzoló karaktere: +

3. téglalap BF sarok X: [0, 78] (negativra vege) 15

3. téglalap BF sarok Y[0, 23] 2

3. téglalap JA sarok X[16, 79] 30

3. téglalap JA sarok Y[3, 24] 7

3. téglalap rajzoló karaktere: -

4. téglalap BF sarok X: [0, 78] (negativra vege) -1

...

## Kimenet (2/2)

```

. . .
| | | | | | | | |
| | | | | | | | |   - - - - -
| | | | | | | | |   - - - - -
| | | | | | | | |   - - - - -
| | | | | | | | |   - - - - -
| | | | | | | | |
| | | | | + + + + + + + + + - - - - -
| | | | | + + + + + + + + + - - - - -
| | | | | + + + + + + + + + +
| | | | | + + + + + + + + + +
| | | | | + + + + + + + + + +
| | | | | + + + + + + + + + +
      + + + + + + + + + +
      + + + + + + + + + +
      + + + + + + + + + +
      + + + + + + + + + +
      + + + + + + + + + +
. . .

```

## teglalap1.cpp

```
1  #include <iostream>
2  using namespace std;
3  #define MAXALAK 128
4  #define MINX 0
5  #define MAXX 79
6  #define MINY 0
7  #define MAXY 24
8
9  struct koordinata {
10     int x, y;
11 };
12
13 struct teglalap {
14     koordinata bf, ja;
15     char c;
16 };
```



## teglalap1.cpp

```

48  int main() {
49      teglalap tt[MAXALAK];
50      int db=0, k;  bool folytat=true;
51      cout << "Rajzprogram – adj meg a téglalap adatait!\n";
52      while(db<MAXALAK and folytat) {
53          do {
54              cout << db+1 << ". teglalap BF sarok X: [" << MINX
55                  << ", " << MAXX-1 << "]" (negativra vege) ";
56              cin >> k;
57              folytat = k>=0;
58          } while(folytat && (k<MINX or k>MAXX-1));
59          if(folytat) {
60              tt[db].bf.x = k;
61              tt[db].bf.y = beker(db+1, "BF sarok Y", MINY, MAXY-1);
62              tt[db].ja.x = beker(db+1, "JA sarok X", tt[db].bf.x+1, MAXX);
63              tt[db].ja.y = beker(db+1, "JA sarok Y", tt[db].bf.y+1, MAXY);
64              cout << db+1 << ". teglalap rajzoló karaktere: ";
65              cin >> tt[db].c;
66              db++;
67          }
68      }
69      rajzol(tt, db);
70      return 0;
71  }

```

## teglalap1.cpp

```
38  int beker(int db, string s, int min, int max) {
39      int k;
40      do {
41          cout << db << ". teglalap " << s << '['
42              << min << ", " << max << "]" << " ";
43          cin >> k;
44      } while(k<min or k>max);
45      return k;
46  }
```

## teglalap1.cpp

```
18 bool takarja(teglalap t, int s, int o) {
19     return (t.bf.x<=o and t.ja.x>=o) and
20            (t.bf.y<=s and t.ja.y>=s);
21 }
22
23 void rajzol(teglalap tt[MAXALAK], int db) {
24     for(int s=MINY; s<=MAXY; s++) {
25         for(int o=MINX; o<=MAXX; o++) {
26             bool takarasban = false;
27             for(int t=db-1; t>=0 and not takarasban; t--) {
28                 if(takarja(tt[t], s, o)) {
29                     cout << tt[t].c; takarasban = true;
30                 }
31             }
32             if(not takarasban) cout << ' ';
33         }
34         cout << endl;
35     }
36 }
```