Programozás
(GKxB_INTM021)

Dr. Hatwágner F. Miklós

Széchenyi István Egyetem, Győr

2018. szeptember 27.

Háromszög szerkeszthetőségének ellenőrzése

haromszog1.cpp Az oldalhossz beolvasása 3x ismétlődik!

```
#include <iostream>
using namespace std;
int main() {
  int a, b, c;
  bool megszerkesztheto = false;
  cout << "Adja meg egy haromszog oldalhosszait!\n";
  do {
    do { // hatultesztelo ciklus eleje...
      cout << "A oldal hossza: ";
      cin >> a;
    } while(a <= 0); // ...es vege
    do {
      cout << "B oldal hossza: ";
      cin >> b;
    } while(b <= 0);
    do {
      cout << "C oldal hossza: ";
      cin >> c;
    } while(c <= 0);
    if(a+b<=c or b+c<=a or c+a<=b) // alternativ szintakszis
      cout << "Ez nem szerkeszthetto meg!\n";
    else {
      megszerkesztheto = true;
      cout << "Megszerkeszthetto.\n"; }
  } while(not megszerkesztheto);
  return 0; }
```

Háromszög szerkeszthetőségének ellenőrzése

haromszog3.cpp

```
#include <iostream>
#define OLDALSZAM 3
using namespace std;
int main() {
  int ot[OLDALSZAM]; // 3 elemu tomb az oldalhosszaknak
  int i;             // aktualis oldal indexe
  bool megszerkesztheto = false;
  cout << "Adja meg egy haromszog oldalhosszait!\n";
  do {
    i = 0;
    while(i < OLDALSZAM) { // Csak 1x irjuk meg a beolvasast!
      do {
        cout << "A kovetkezo oldal hossza: ";
        cin >> ot[i];                          // tomb indexeles
      } while(ot[i] <= 0);
      i++;
    }
    if(ot[0]+ot[1]<=ot[2] or ot[1]+ot[2]<=ot[0] or ot[2]+ot[0]<=ot[1])
      cout << "Ez nem szerkesztheto meg!\n";
    else {
      megszerkesztheto = true;
      cout << "Megszerkesztheto.\n"; }
  } while(not megszerkesztheto);
  return 0; }
```

Háromszög szerkeszthetőségének ellenőrzése

Tömb definíció

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

Tömb tárigénye

sizeof(tömbazonosító) $\equiv$ *méret*sizeof(típus)*

Tömbelemek (indexes változó) elérése

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

Háromszög szerkeszthetőségének ellenőrzése

haromszog4.cpp

```
#include <iostream>
#define OLDALSZAM 3
using namespace std;
int main() {
  int ot[OLDALSZAM];
  int i;
  bool megszerkesztheto = false;
  char onev;                          // aktualis oldal megnevezese
  cout << "Adja meg egy haromszog oldalhosszait!\n";
  do {
    i = 0; onev = 'A';
    while(i < OLDALSZAM) {
      do {
        cout << onev << " oldal hossza: "; // oldal megnevezese
        cin >> ot[i];
      } while(ot[i] <= 0);
      i++; onev++;                    // kovetkezo oldal neve
    }
    if(ot[0]+ot[1]<=ot[2] or ot[1]+ot[2]<=ot[0] or ot[2]+ot[0]<=ot[1])
      cout << "Ez nem szerkesztheto meg!\n";
    else {
      megszerkesztheto = true;
      cout << "Megszerkesztheto.\n"; }
  } while(not megszerkesztheto);
  return 0; }
```

Háromszög szerkeszthetőségének ellenőrzése

Oldal nevének előállítása

- 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 $\rightarrow$ ASCII kód: `'0'+szamjegy`
- ASCII kód $\rightarrow$ Számjegy: `karakter-'0'`
- Betűk is hasonlóan kezelhetők

Számjegy karakterek számlálása

szamlalo2.cpp 1/2

```
1  #include <iostream>
2  #include <cstdio>
3  using namespace std;
4
5  int main(void) {
6    cout << "Szamjegyek, ures- es egyeb karakterek leszamlalasa\n"
7         << "a bemeneten EOF-ig vagy Ctrl+D-ig.\n\n";
8    int k, feher=0, egyeb=0;
9    int nulla=0, egy=0, ketto=0, harom=0, negy=0, // :(
10       ot=0, hat=0, het=0, nyolc=0, kilenc=0;
11   while((k=cin.get()) != EOF){
12     switch(k) {                          // Ronda, mint egy loharapas!
13       case '0': nulla++; break;
14       case '1': egy++; break;
15       case '2': ketto++; break;
16       case '3': harom++; break;
17       case '4': negy++; break;
18       case '5': ot++; break;
19       case '6': hat++; break;
20       case '7': het++; break;
21       case '8': nyolc++; break;
22       case '9': kilenc++; break;
23       case ' ': case '\n': case '\t': feher++; break;
24       default: egyeb++; break;
25     }
26   }
```

Számjegy karakterek számlálása

szamlalo2.cpp 2/2

```
27      cout << "Szamjegyek:\n";
28      cout << "Nulla:\t" << nulla << " db\n"; // Uram irgalmazz!
29      cout << "Egy:\t" << egy << " db\n";
30      cout << "Ketto:\t" << ketto << " db\n";
31      cout << "Harom:\t" << harom << " db\n";
32      cout << "Negy:\t" << negy << " db\n";
33      cout << "Ot:\t" << ot << " db\n";
34      cout << "Hat:\t" << hat << " db\n";
35      cout << "Het:\t" << het << " db\n";
36      cout << "Nyolc:\t" << nyolc << " db\n";
37      cout << "Kilenc:\t" << kilenc << " db\n";
38      cout << "Ures karakterek: " << feher << ", egyeb: " << egyeb << endl;
39      return 0;
40  }
```

Nyilvánvalóan szükségünk van egy tömbre!

Számjegy karakterek számlálása

szamlalo3.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 "
8         << "leszamlalasa\na bemeneten EOF-ig "
9         << "vagy Ctrl+D-ig.\n\n";
10   int i, k, feher=0, egyeb=0;
11   int szamjegy[DB];  // 10 elemu tomb a tiz szamjegynek
12   i = 0;
13   while(i < DB) {
14     szamjegy[i] = 0; // Szamlalok nullazasa
15     i++;
16   }
```

Számjegy karakterek számlálása

szamlalo3.cpp 2/2

```
17    while((k=cin.get()) != EOF){
18      if(k>='0' and k<='9') {
19        i = k-'0';       // Karakter atalakitasa szamma,
20        szamjegy[i]++; // amit indexkent hasznalunk
21      } else if(k==' ' or k=='\n' or k=='\t') feher++;
22      else egyeb++;
23    }
24    cout << "Szamjegyek:\n";
25    i = 0;                  // Eredmenyek kijelzese
26    while(i < DB) {
27      cout << i << '\t' << szamjegy[i] << " db\n";
28      i++;
29    }
30    cout << "Ures karakterek: " << feher
31         << ", egyeb: " << egyeb << endl;
32    return 0;
33  }
```

Számjegy karakterek számlálása

Tömbelemek, mint számlálók

Az i számjegy darabszámát szamjegy[i] tárolja! (Azaz pl. 0-ból szamjegy[0], 1-ből szamjegy[1], sít. 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 specifikálták, a fordító megállapítja *inicializátorlista* elemszámából
- De a *<méret>* és az *inicializátorlista* közül legalább az egyiknek léteznie kell!

Számjegy karakterek számlálása

szamlalo4.cpp

```
#include <iostream>
#include <cstdio>
#define DB 10
using namespace std;

int main(void) {
  cout << "Szamjegyek, ures- es egyeb karakterek leszamlalasa\n"
       << "a bemeneten EOF-ig vagy Ctrl+D-ig.\n\n";
  int k, i, feher=0, egyeb=0;
  int szamjegy[DB] = {0}; // szamlalok nullazasa inicializalassal
  while((k=cin.get()) != EOF){
    if(k>='0' && k<='9') ++szamjegy[k-'0']; // szamlalok leptetese
    else if(k==' ' or k=='\n' or k=='\t') ++feher;
    else ++egyeb;
  }
  cout << "Szamjegyek:\n";
  i = 0;
  while(i < DB) {
    cout << i << '\t' << szamjegy[i] << " db\n";
    i++;
  }
  cout << "Ures karakterek: " << feher
       << ", egyeb: " << egyeb << endl;
  return 0;
}
```

Számok kiírása fordított sorrendben

forditva1.cpp

```
#include <iostream>
#define N 5
using namespace std;

int main() {
  cout << "Adjon meg " << N << " szamot, kiirjuk oket forditott "
       << "sorrendben!\n\n";
  int szamok[N], db=0;
  while(db < N) {
    cout << db+1 << ". szam: ";
    cin >> szamok[db];
    db++;
  }
  cout << "\nForditott sorrendben:\n";
  db = N-1;
  while(db >= 0) {
    cout << szamok[db] << '\t';
    db--;
  }
  cout << endl;
  return 0;
}
```

Számok kiírása fordított sorrendben

forditva2.cpp

```
#include <iostream>
#define N 5
using namespace std;

int main() {
  cout << "Adjon meg " << N << " szamot, kiirjuk oket "
       << "forditott sorrendben!\n\n";
  int szamok[N], db=0;
  while(db < N) {
    cout << db+1 << ". szam: ";
    cin >> szamok[db++];                    // osszevonas
  }
  cout << "\nForditott sorrendben:\n";
  while(db--) cout << szamok[db] << '\t'; // osszevonas
  cout << endl;
  return 0;
}
```

binker1.cpp 1/4

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

binker2.cpp 2/4

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

binker3.cpp 3/4

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

binker4.cpp 4/4

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

Buborék rendezés

buborek.cpp

```
#include <iostream>
using namespace std;

int main() {
  int szamok[] = {12, 3, 54, -4, 56, 4, 7, 3};
  int n = sizeof(szamok)/sizeof(szamok[0]); // Tomb elemszam szamolasa
  int i=n-1, k;
  while(i>=1) {
    k = 0;
    while(k<i) {
      if(szamok[k]>szamok[k+1]) {
        int csere = szamok[k];
        szamok[k] = szamok[k+1];
        szamok[k+1] = csere;
      }
      k++;
    }
    i--;
  }
  cout << "Rendezes utan:\n"; i = 0;
  while(i < n) {
    cout << szamok[i] << '\t'; i++;
  }
  cout << endl;
  return 0;
}
```

Az std::string típus

string.cpp

```
#include <iostream>
using namespace std;

int main(void) {
  string s1;
  string s2 = " es Bolka";                    // inicializalas
  cout << "s1: " << s1 << '\n';               // ures string!
  s1 = "Lolka";       // literal ertekul adhato stringnek
  s1 += s2;                                  // stringek osszefuzhetoek
  cout << s1 << "\nAdjon meg egy szot! ";
  cin >> s2;                                 // Szo beolvasasa stringbe
  cout << "A szo hossza: " << s2.length() << '\n';
  // stringek osszehasonlithatoak relacios operatorokkal
  if(s2 == "Frakk") cout << "Megint a regi mese\n";
  cout << s2 << " elso betuje: " << s2[0] << endl;
  return 0;
}
```

Konvertálás kettesből tízes számrendszerbe

kettes1.cpp

```cpp
#include <iostream>
using namespace std;

int main() {
  string b;
  unsigned d, i;
  cout << "Adjon meg egy kettes szamrendszerbeli szamot!\n";
  cin >> b;
  d = i = 0;
  while(i < b.length()) {
    d = d*2 + b[i]-'0';
    i++;
  }
  cout << "Tizes szamrendszerben: " << d << endl;
  return 0;
}
```

Konvertálás tízesből kettes számrendszerbe

kettes2.cpp

```
#include <iostream>
using namespace std;
int main() {
  char b[100];
  int d, i;
  cout << "Adjon meg egy tizes szamrendszerbeli szamot!\n";
  cin >> d;
  i = 0;
  while(d > 0) {
    b[i] = d%2+'0'; d /= 2; i++;
  }
  cout << "Kettes szamrendszerben: ";
  i--;
  while(i >= 0) {
    cout << b[i]; i--;
  }
  cout << endl;
  return 0;
}
```

Neptun kód ellenőrzés

neptun1.cpp

```
#include <iostream>
using namespace std;
int main(void) {
  bool helytelen;
  string neptun;          // karakterlanc tarolasara szolgalo C++ tipus
  do {
    helytelen = false;
    cout << "Adja meg a Neptun kodjat: "; cin >> neptun;
    if(neptun.length() != 6) { // karakterlanc hosszanak lekerdezese
      cout << "Hat karakterbol kell allnia!\n"; helytelen = true;
    } else {
      unsigned i=0;
      while(not helytelen and i<neptun.length()) {
        char k = neptun[i];
        bool szamjegy = k>='0' and k<='9';
        bool nagybetu = k>='A' and k<='Z';
        bool kisbetu  = k>='a' and k<='z';
        if(not szamjegy and not nagybetu and not kisbetu) {
          cout << "Csak szamjegyeket es betuket tartalmazhat!\n";
          helytelen = true; }
        i++; } }
  } while(helytelen);
  cout << "Rendben.\n";
  return 0; }
```

Neptun kód ellenőrzés

neptun2.cpp

```
#include <iostream>
#include <cctype> // toupper() miatt
using namespace std;
int main(void) {
  bool helytelen;
  string neptun;
  do {
    helytelen = false;
    cout << "Adja meg a Neptun kodjat: "; cin >> neptun;
    if(neptun.length() != 6) { // karakterlanc hosszanak lekerdezese
      cout << "Hat karakterbol kell allnia!\n"; helytelen = true;
    } else {
      unsigned i=0;
      while(not helytelen and i<neptun.length()) {
        char k = toupper(neptun[i]);          // nagybeture alakitas
        if((k<'0' or k>'9') and (k<'A' or k>'Z')) {
          cout << "Csak szamjegyeket es betuket tartalmazhat!\n";
          helytelen = true; }
        i++; } }
  } while(helytelen);
  cout << "Rendben.\n";
  return 0; }
```

Neptun kód ellenőrzés

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 értékként kezelendő

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

Neptun kód ellenőrzés

neptun3.cpp

```
#include <iostream>
#include <cctype> // isalnum() miatt
using namespace std;
int main(void) {
  bool helytelen;
  string neptun;
  do {
    helytelen = false;
    cout << "Adja meg a Neptun kodjat: "; cin >> neptun;
    if(neptun.length() != 6) { // karakterlanc hosszanak lekerdezese
      cout << "Hat karakterbol kell allnia!\n"; helytelen = true;
    } else {
      unsigned i=0;
      while(not helytelen and i<neptun.length()) {
        if(not isalnum(neptun[i])) { // alfanumerikus karakter?
          cout << "Csak szamjegyeket es betuket tartalmazhat!\n";
          helytelen = true; }
        i++; } }
  } while(helytelen);
  cout << "Rendben.\n";
  return 0; }
```