Programozás
(GKxB_INTM021)

Dr. Hatwágner F. Miklós

Széchenyi István Egyetem, Győr

2018. február 18.

Minimum és maximumkeresés

minmax1.cpp

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

int main() {
  cout << "Adjon meg nem negativ egesz szamokat,\n"
       << "megkeressuk kozottuk a minimalisat es a maximalisat.\n"
       << "Kilepes negativ szam megadasaval.\n";
  int db=0, akt=1; // inicializacio
  int min, max;
  while(akt >= 0) {
    cout << "Kovetkezo szam: ";
    cin >> akt;
    if(akt >= 0) {
      if(db == 0) min = max = akt; // tobbszoros hozzarendeles
      else if(akt > max) max = akt;
      else if(akt < min) min = akt;
      db++; // noveles eggyel
    }
  }
  if(db > 0) cout <<   "A minimum: " << min
                  << "\nA maximum: " << max << '\n';
  else cout << "Nem adott meg adatokat.\n";
  return 0;
}
```

Minimum és maximumkeresés

Változó

deklaráció típus és azonosító megadása, helye: felhasználáskor vagy előtte

definíció deklaráció + memóriaterület foglalása

inicializáció definíciókor kezdőérték megadása, pl. `int db=0;`

= operátor (hozzárendelés op.)

- asszociativitás: jobbról balra
- `min = max = akt;` ≡ `max = akt; min = max;`

Többirányú elágazás: *if(...) ... else if(...) ... else if(...) ... else ...*

Minimum és maximumkeresés

Növelő és csökkentő operátorok

$++$ növelés eggyel

$--$ csökkentés eggyel

Létezik elő- és utótag (prefix/postfix) alak is $\rightarrow$ műveleti sorrend!

Az előtag/utótag operátorok hatása az eredményre

```
int a, b; // a és b értéke definiálatlan
b = 6;    // b értéke mostantól 6
a = ++b;  // 1) b értéke nő 7-re,
          // 2) ezt hozzárendeli a-hoz
a = b++;  // 1) hozzárendeli b értékét a-hoz,
          // 2) növeli b értékét 8-ra
```

Minimum és maximumkeresés

minmax2.cpp

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

int main() {
  cout << "Adjon meg nem negativ egesz szamokat,\n"
       << "megkeressuk kozottuk a minimalisat es a maximalisat.\n"
       << "Kilepes negativ szam megadasaval.\n";
  int db=0, akt; // akt-nak nem kell kezdoertek
  int min, max;
  cout << "Kovetkezo szam: "; // kod elso elofordulasa
  cin >> akt;
  while(akt >= 0) {
    if(db == 0) min = max = akt; // innen viszont eltunt egy feltetel
    else if(akt > max) max = akt;
    else if(akt < min) min = akt;
    db++;
    cout << "Kovetkezo szam: "; // kod masodik elofordulasa
    cin >> akt;
  }
  if(db > 0) cout <<   "A minimum: " << min
                  << "\nA maximum: " << max << '\n';
  else cout << "Nem adott meg adatokat.\n";
  return 0;
}
```

Pozitív szám beolvasása

pozitiv1.cpp

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

int main() {
  int szam;
  cout << "Adjon meg egy pozitiv szamot! "; // uzenet
  cin >> szam;                              // beolvasas
  while(szam <= 0) { // ismetles, ha hibas volt az input
    cout << "Adjon meg egy pozitiv szamot! ";
    cin >> szam;
  }
  cout << "Beolvasott ertek: " << szam << endl;
  return 0;
}
```

Pozitív szám beolvasása

pozitiv2.cpp

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

int main() {
  int szam = -1;      // az inicializacio kikenyszeriti
  while(szam <= 0) { // a ciklusmag futasat
    cout << "Adjon meg egy pozitiv szamot! ";
    cin >> szam;
  }
  cout << "Beolvasott ertek: " << szam << endl;
  return 0;
}
```

Pozitív szám beolvasása

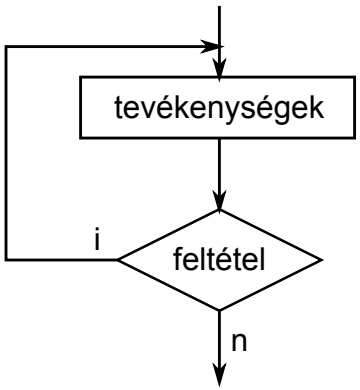
pozitiv3.cpp

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

int main() {
  int szam; // nincs szukseg inicializaciora
  do {      // a ciklusmag 1+ futtatasahoz
    cout << "Adjon meg egy pozitiv szamot! ";
    cin >> szam;
  } while(szam <= 0);
  cout << "Beolvasott ertek: " << szam << endl;
  return 0;
}
```

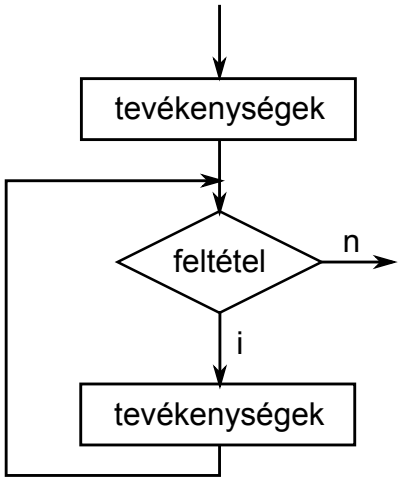
Háromszög szerkeszthetőségének ellenőrzése

Hátultesztelő ciklus – a ciklusmag egyszer biztosan lefut




```
do {
    tevékenységek
} while(feltétel kifejezése);
```

Ciklusmag futásának kikényszerítése elöltesztelő ciklussal




```
tevékenységek
while(feltétel kifejezése) {
    tevékenységek
}
```

Háromszög szerkeszthetőségének ellenőrzése

haromszog1.cpp

```
#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 szerkeszthetо meg!\n";
    else {
      megszerkesztheto = true;
      cout << "Megszerkeszthetо.\n"; }
  } while(not megszerkesztheto);
  return 0; }
```

Háromszög szerkeszthetőségének ellenőrzése

Logikai operátorok

- `!`, `not`: logikai nem, tagadás
- `||`, `or`: logikai (megengedő) vagy
- `&&`, `and`: logikai és

Igazságtáblázat

a	b	not a	a or b	a and b
false	false	true	false	false
false	true	true	true	false
true	false	false	true	false
true	true	false	true	true

(Rész)kifejezések kiértékelésének optimalizálása (short-circuit evaluation)

Háromszög szerkeszthetőségének ellenőrzése

haromszog2.cpp

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

int main() {
  int a, b, c;
  cout << "Adja meg egy haromszog oldalhosszait novekvo sorrendben!\n";
  do {
    cout << "A oldal hossza: ";
    cin >> a;
  } while(a <= 0);
  do {
    cout << "B oldal hossza: ";
    cin >> b;
  } while(b < a);
  do {
    cout << "C oldal hossza: ";
    cin >> c;
  } while(c<b or a+b<=c);
  return 0;
}
```

Körlemez rajzolása

kor1.cpp

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

int main() {
  int sor = -5; // A kor sugara 5
  while(sor <= 5) {
    int oszlop = -5;
    while(oszlop <= 5) {
      if(5*5 >= sor*sor + oszlop*oszlop) cout << '*';
      else cout << ' ';
      oszlop++;
    }
    sor++;
    cout << '\n';
  }
  return 0;
}
```

Körlemez rajzolása

Problémák:

- a kurzor pozicionálása korlátozott
- rengeteg helyen szerepel ugyanaz a konstans: nehézkes módosítás, hibalehetőségek
- a karakterek kb. 2x magasabbak, mint amilyen szélesek

Kimenet

```
      *
   *******
  *********
  *********
  *********
  *********
***********
  *********
  *********
  *********
  *********
   *******
      *
```

Körlemez rajzolása

kor2.cpp

```
#include <iostream>
#define R 10 // A kor sugara
using namespace std;

int main() {
  int sor = -R;
  while(sor <= R) {
    int oszlop = -R;
    while(oszlop <= R) {
      if(R*R >= sor*sor + oszlop*oszlop) cout << '*';
      else cout << ' ';
      oszlop++;
    }
    sor += 2; // Noveles kettovel
    cout << '\n';
  }
  return 0;
}
```

Körlemez rajzolása

`#define`

- szimbolikus állandók, egyszerű makrók
- előfeldolgozó „egyszerű” szöveghelyettesítést végez
- Nincs pontosvessző a végén!

Összevont operátorok

- `sor += 2;` ≡ `sor = sor+2;`
- $+=$, $-=$, $*=$, $/=$, $\%=$

Egyoperandusos $+$ és $-$ operátorok

Kimenet

```
          *
     ************
   ****************
  ******************
  ******************
*********************
  ******************
  ******************
   ****************
     ************
          *
```

Betűk, szavak, sorok számlálása

szamlalo.cpp

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

int main(void) {
  int k, sorDb, szoDb, karDb;
  bool szoban = false;
  cout << "A bemenet karaktereinek, sorainak és\n"
       << "szavainak leszámlálása\n"
       << "A bemenet vége: Ctrl+D vagy EOF.\n\n";
  sorDb = szoDb = karDb = 0;
  while((k=cin.get()) != EOF){
    ++karDb;
    if(k == '\n') ++sorDb;
    if(k==' ' or k=='\n' or k=='\t') szoban = false;
    else if(not szoban){
      szoban = true;
      ++szoDb;
    }
  }
  cout << "sor = " << sorDb << ", szo = " << szoDb
       << ", karakter = " << karDb << endl;
  return 0;
}
```

Betűk, szavak, sorok számlálása

Egy karakter beolvasása: `int get()`
Karakter ábrázolása int típusban
Bemenet vége: `EOF` → `<cstdio>`
Műveleti sorrend! `while((k=cin.get()) != EOF){`
A `szoban` szerepe

Operátorok precedenciája és asszociativitása

Operátor	Asszociativitás
a++ a−−	balról jobbra
++a −−a +a −a ! sizeof	jobbról balra
a*b a/b a%b a+b a−b < <= > >= == != && ||	balról jobbra
= += −= *= /= %=	jobbról balra
,	balról jobbra

Angol sorszámnevek

ordinal1.cpp

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

int main() {
  cout << "Szam: ";
  int szam;
  cin >> szam;
  if(szam == 0) cout << '0';
  else {
    cout << szam;
    if(szam>10 and szam<21) cout << "th";
    else if(szam%10 == 1) cout << "st";
    else if(szam%10 == 2) cout << "nd";
    else if(szam%10 == 3) cout << "rd";
    else cout << "th";
  }
}
```

Problémák: nagyon sok irányú elágazás, felesleges osztások

Angol sorszámnevek

ordinal2.cpp

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

int main() {
  cout << "Szam: ";
  int szam;
  cin >> szam;
  if(szam == 0) cout << '0';
  else {
    cout << szam;
    if(szam>10 and szam<21) cout << "th";
    else switch(szam%10) {
      case 1: cout << "st"; break;
      case 2: cout << "nd"; break;
      case 3: cout << "rd"; break;
      default: cout << "th";
    }
  }
}
```

Angol sorszámnevek

- switch(*kifejezés*) *utasítás*
- *kifejezés* egész típusú
- *utasítás* tartalmazhat
  - több case *konstans-kifejezés: utasítás*-t,
  - nulla vagy egy default: *utasítás*-t
- végrehajtás leáll:
  - switch blokkjának végén
  - az első break utasításnál
- *konstans-kifejezés* egész típusú
- *kifejezés* és *konstans-kifejezés* értékeinek összehasonlítása
- több case címke is címkézheti ugyanazt az utasítást, de minden címkének egyedinek kell lennie
- switch utasítások egymásba ágyazhatóak

minmax2.cpp (Emlékeztető)

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

int main() {
  cout << "Adjon meg nem negativ egesz szamokat,\n"
       << "megkeressuk kozottuk a minimalisat es a maximalisat.\n"
       << "Kilepes negativ szam megadasaval.\n";
  int db=0, akt; // akt-nak nem kell kezdoertek
  int min, max;
  cout << "Kovetkezo szam: "; // kod elso elofordulasa
  cin >> akt;
  while(akt >= 0) {
    if(db == 0) min = max = akt; // innen viszont eltunt egy feltetel
    else if(akt > max) max = akt;
    else if(akt < min) min = akt;
    db++;
    cout << "Kovetkezo szam: "; // kod masodik elofordulasa
    cin >> akt;
  }
  if(db > 0) cout <<   "A minimum: " << min
                  << "\nA maximum: " << max << '\n';
  else cout << "Nem adott meg adatokat.\n";
  return 0;
}
```

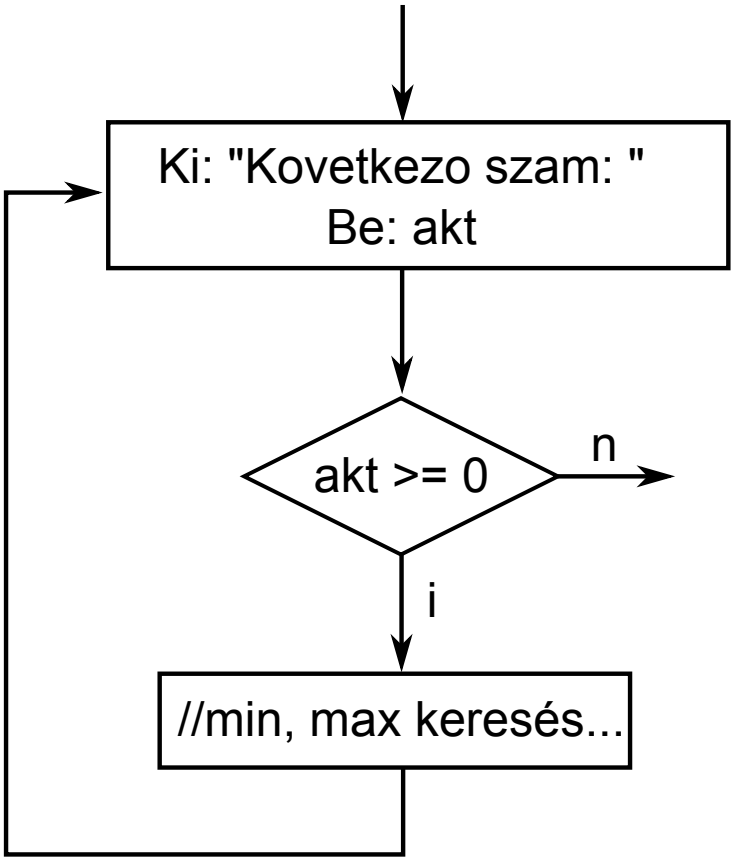
Minimum és maximumkeresés

minmax3.cpp

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

int main() {
  cout << "Adjon meg nem negativ egesz szamokat,\n"
       << "megkeressuk kozottuk a minimalisat es a maximalisat.\n"
       << "Kilepes negativ szam megadasaval.\n";
  int db=0, akt;
  int min, max;
  while(cout<<"Kovetkezo szam: ", cin>>akt, akt>=0) { // , operator
    if(!db) min = max = akt; // ! operator
    else if(akt > max) max = akt;
    else if(akt < min) min = akt;
    db++;
  }
  if(db) cout <<   "A minimum: " << min // logikai kifejezes
              << "\nA maximum: " << max << '\n';
  else cout << "Nem adott meg adatokat.\n";
  return 0;
}
```

Minimum és maximumkeresés



Vessző operátor

- összetett, külön-külön is értelmes kifejezésekből álló kifejezés szerepeltethető ott, ahol csak egy kifejezés állhat
- kifejezés értéke = az utolsó részkifejezés értéke

Minimum és maximumkeresés

Logikai kifejezések

- `bool` típus
- `false` $\equiv$ 0
- `true` $\equiv$ 1
- nulla értékű egész $\rightarrow$ hamis
- nem nulla értékű egész $\rightarrow$ igaz
- `if(db) ...` $\equiv$ `if(db != 0) ...`
- `if(!db) ...` $\equiv$ `if(db == 0) ...`