

Programozás

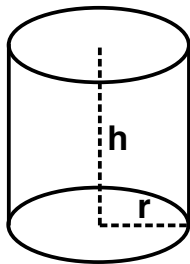
(GKxB_INTM114)

Dr. Hatwagner F. Miklós

Széchenyi István Egyetem, Győr

https://github.com/wajzy/GKxB_INTM114.git

2024. március 4.



Feladat:

- 1 Beolvasandó a henger magassága és alapjának sugara
- 2 Kiszámítandó a henger felszíne, térfogata

$$V = r^2 \pi h$$

$$S = 2r\pi h + 2r^2\pi = 2r\pi(r + h)$$

Típus	Méret	Számábrázolási határok	Pontosság
float	4 bájt	$\pm 3,4 \cdot 10^{-38} - \pm 3,4 \cdot 10^{+38}$	6-7 dec. jegy
double	8 bájt	$\pm 1,7 \cdot 10^{-308} - \pm 1,7 \cdot 10^{+308}$	15-16 dec. jegy
long double	10 bájt	$\pm 1,2 \cdot 10^{-4932} - \pm 1,2 \cdot 10^{+4932}$	19 dec. jegy

henger.cpp

```
1 #define _USE_MATH_DEFINES // regi rendszerekhez
2 #include <iostream>
3 #include <cmath> // vagy math.h
4 using namespace std;
5
6 int main() {
7     double r, h;
8     cout << "Adja meg egy henger\n\talapjának sugarát! "; cin >> r;
9     cout << "\tmagasságát! "; cin >> h;
10    cout << "A henger\n\tterfogata: " << r*r*M_PI*h
11        << "\n\tfelszíne: " << 2.*r*M_PI*(r+h) << endl;
12    return 0;
13 }
```

Lebegőpontos konstansok főbb tulajdonságai

- ábrázolási határok → `cfloat` (`float.h`), pl.
 - `DBL_MIN` a `double` típusal ábrázolható legkisebb pozitív szám
 - `DBL_MAX` a `double` típusal ábrázolható legnagyobb véges szám
- mantisszából elhagyható az egész- vagy a törtrész, de **mindkettő nem!**
- elhagyható a tizedes pont vagy az exponens (`e`, `E`) rész, de **mindkettő nem!**
- utótag nélkül a belső ábrázolás `double`

- Utótagok a konstans belső ábrázolásának megváltoztatásához:
 - f, F (float)
 - l, L (long double)

Néhány lebegőpontos konstans

-5., .3, 5.3, -5e4, 5.67E-12, -1.23e-4l, 5.F

A `cmath` (`math.h`) néhány (nem feltétlenül szabványosított) konstansa

- `M_E` – Euler-konstans
- `M_PI` – π értéke
- `M_SQRT2` – $\sqrt{2}$

Az **egész** konstansok főbb tulajdonságai

- megadható decimális, oktális (0...) és hexadecimális (0x..., 0X...) alakban
- **utótagok** a konstans típusának megváltoztatásához:
 - u, U (unsigned)
 - l, L (long)
 - ll, LL (long long)
 - z, Z (std::size_t)

Egész típusú változók és konstansok

```
int i = 1;                unsigned ui = 8u;  
int j = 010; /* == 8 */   long li = 16L;  
int k = 0x2A; /* == 42 */ unsigned long uli = 666Ul;
```

- platformfüggő méretű egész típusok ábrázolási korlátai → `climits` (`limits.h`)
- rögzített szélességű egész típusok → `cstdint` (`stdint.h`).

`climits` részletek

```
#define SCHAR_MIN (-128)
#define UCHAR_MAX 255
#define SHRT_MAX 32767
#define INT_MAX 2147483647
#define ULONG_MAX 18446744073709551615UL
```

masodfoku.cpp Másodfokú egyenlet megoldóképlete: $x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

```

1  #include <iostream>
2  #include <cmath> // sqrt() miatt
3  using namespace std;
4
5  int main() {
6      double a, b, c;
7      cout << "Az ax^2+bx+c = 0 egyenlet megoldasa\n";
8      cout << "a erteke: "; cin >> a;
9      if(a == 0.) {
10         cout << "Az egyenlet nem masodfoku!\n";
11     } else {
12         cout << "b erteke: "; cin >> b;
13         cout << "c erteke: "; cin >> c;
14         double d = b*b - 4.*a*c;
15         if(d < 0.) {
16             cout << "Nincs valos gyok!\n";
17         } else {
18             cout << "x1 = " << (-b + sqrt(d)) / (2.*a)
19                 << "\nx2 = " << (-b - sqrt(d)) / (2.*a) << endl;
20         }
21     }
22     return 0;
23 }

```


- Szabványos függvénykönyvtárak → hordozhatóság
- Beszerkesztendő fejléc: `cmath` (`math.h`)
- Függvények paramétereinek típusa és visszatérési értéke többnyire `double`
- Trigonometrikus fv.-ek paramétere és visszatérési értéke **radiánban** értendő

Néhány gyakran használt matematikai függvény

Prototípus	Funkció
<code>double ceil(double x)</code>	Az x-nél nagyobb egészek közül a legkisebbet adja
<code>double cos(double x)</code>	koszinusz
<code>double cosh(double x)</code>	hiperbolikus koszinusz
<code>double exp(double x)</code>	exponenciális fv.
<code>double fabs(double x)</code>	abszolút érték
<code>double fmod(double x, double y)</code>	osztás lebegőpontos maradékát adja
<code>double log(double x)</code>	természetes alapú logaritmus
<code>double log10(double x)</code>	10-es alapú logaritmus
<code>double pow(double x, double y)</code>	hatványozás
<code>double sqrt(double x)</code>	négyzetgyökvonás

haromszog4.cpp

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

haromszog5.cpp

```
9      int i = 0; // Nincs külön változója az oldal nevenek
10     while(i < OLDALSZAM) {
11         do {
12             cout << 'A'+i << " oldal hossza: "; // Osszeg tipusa?
13             cin >> ot[i];
14         } while(ot[i] <= 0.);
15         i++;
16     }
```

Kimenet

Adja meg egy haromszog oldalhosszait!

65 oldal hossza:

Implicit típuskonverzió: kétoperandusos műveleteknél, ha az operandusok típusa eltér. Általában a „pontosabb” típusra alakít. Szabályok:

egyik operandus	másik operandus
long double	<i>bármilyen</i> → long double
double	<i>bármilyen</i> → double
float	<i>bármilyen</i> → float
egész-előléptetés	egész-előléptetés
unsigned long	<i>bármilyen</i> → unsigned long
long → (unsigned) long	unsigned int → (unsigned) long
long	<i>bármilyen</i> → long
unsigned int	<i>bármilyen</i> → unsigned int
int	int

Egész-előléptetés (integral promotion)

Régi típus	Új típus	Átalakítási módszer
char	int	Alapértelmezett (signed/unsigned) char típustól függően.
unsigned char	int	Felső bájtok feltöltése zérus bitekkel.
signed char	int	Előjel kiterjesztése a felső bájtokra.
short int	int	Előjel kiterjesztés.
unsigned short	unsigned int	Feltöltés zérus bitekkel.

Figyelem!

- A konverzió időigényes!
- Karakterlánc sosem alakul aritmetikai értékké!

haromszog6.cpp

```
9      int i = 0;
10     while(i < OLDALSZAM) {
11         do {
12             cout << char('A'+i) << " oldal hossza: ";
13             //cout << (char)('A'+i) << " oldal hossza: ";
14             cin >> ot[i];
15         } while(ot[i] <= 0.);
16         i++;
17     }
```

Kimenet

Adja meg egy haromszog oldalhosszait!

A oldal hossza:

fahrCels1.cpp

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     cout << "Fahrenheit —> Celsius\n"
6         << "Fahrenheit: ";
7     double f;
8     cin >> f;
9     cout << "Celsius: "
10    // Egeszosztas, implicit tip.konv.
11        << (5/9)*(f-32) << endl;
12     return 0;
13 }
```

Fahrenheit – Celsius átváltás

$$C = \frac{5}{9}(F - 32)$$

Kimenet

Fahrenheit -> Celsius
Fahrenheit: 72
Celsius: 0

Explicit típuskonverzió (Type cast)

fahrCels2.cpp

```
10 // Implicit tip.konv.
11 << (5./9)*(f-32) << endl;
```

Kimenet

Celsius: 22.2222

fahrCels4.cpp

```
10 // Nincs típuskonverzió
11 << (5./9.)*(f-32.) << endl;
```

Kimenet

Celsius: 22.2222

fahrCels3.cpp

```
10 // Explicit, implicit tip.konv.
11 << (double(5)/9)*(f-32.)
```

Kimenet

Celsius: 22.2222

fahrCels5.cpp

```
10 // Ertelmetlen explicit tip.konv.
11 << double(5/9)*(f-32.)
```

Kimenet

Celsius: 0

Feltételes operátor: **?:**

Szelekciós tevékenység

```
if(logikaiKifejezes) {  
    valtozo = ertekHalgaz;  
} else {  
    valtozo = ertekHaHamis;  
}
```

Feltételes operátor

```
valtozo = logikaiKifejezes ? ertekHalgaz : ertekHaHamis;
```

haromszog7.cpp

```
18 megszerkesztheto = (ot[0]+ot[1]>ot[2] and ot[1]+ot[2]>ot[0] and ot[2]+ot[0]>ot[1]);
19 cout << (megszerkesztheto ? "Megerszerkesztheto.\n" : "Ez nem szerkesztheto meg!\n");
```

abszolot1.cpp

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     double v, abs;
6     cout << "Szam: "; cin >> v;
7     cout << "Abszolot erteke: ";
8
9     if(v < 0.) abs = -v;
10    else abs = v;
11
12    cout << abs;
13    return 0;
14 }
```

abszolot2.cpp

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     double v, abs;
6     cout << "Szam: "; cin >> v;
7     cout << "Abszolot erteke: ";
8
9     abs = v < 0. ? -v : v;
10
11    cout << abs;
12    return 0;
13
14 }
```

Operátorok precedenciája és asszociativitása

Operátor	Asszociativitás
<code>a++ a--</code> <code>type() fn() array[]</code>	balról jobbra
<code>++a --a</code> <code>+a -a</code> <code>!</code> <code>(type)</code> <code>sizeof</code>	jobbról balra
<code>a*b a/b a%b</code> <code>a+b a-b</code> <code>< <= > >=</code> <code>== !=</code> <code>&&</code> <code> </code>	balról jobbra
<code>a?b:c</code> <code>= += -= *= /= % =</code>	jobbról balra
<code>,</code>	balról jobbra

for (növekményes) ciklus

for(<init-kifejezés>; <kifejezés>; <léptető-kifejezés>) utasítás

- 1 *init-kifejezés* végrehajtása, ha megadták
- 2 *utasítás* végrehajtása, ha *kifejezés* igaz; lehet összetett
- 3 *léptető-kifejezés* végrehajtása, ha megadták, majd ugrás a 2. pontra

Tipikus forgatókönyv:

while

```
ciklusValtozo = kezdErtek;  
while(ciklusValtozo < vegErtek) {  
    ciklusMag;  
    ciklusValtozo += lepes;  
}
```

for

```
for(ciklusValtozo=kezdErtek; ciklusValtozo<vegErtek; ciklusValtozo += lepes) {  
    ciklusMag;  
}
```

N darab szám beolvasása, tárolása, kiírása fordított sorrendben

forditva1.cpp

```
7   int szamok[N], db=0;
8   while(db < N) {
9       cout << db+1 << ". szam: ";
10      cin >> szamok[db++];
11  }
12  cout << "\nMegforditva:\n";
13  while(db-->0) {
14      cout << szamok[db] << '\t';
15  }
```

forditva3.cpp

```
7   int szamok[N], db;
8   for(db=0; db<N; db++) {
9       cout << db+1 << ". szam: ";
10      cin >> szamok[db];
11  }
12  cout << "\nMegforditva:\n";
13  for(db=N-1; db>=0; db--) {
14      cout << szamok[db] << '\t';
15  }
```

Tíz-es számrendszerbeli szám átalakítása kettes számrendszerbelivé

kettes2.cpp

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      string b; char c; int d;
5      cout << "Adjon meg egy tízes "
6           << "számrendszerbeli számot!\n";
7      cin >> d;
8      while(d > 0) {
9          c = d%2 + '0'; b = c + b; d /= 2;
10     }
11     cout << "Kettes számrendszerben: " << b
12          << endl;
13     return 0;
14 }
```

kettes3.cpp

```
1  #include <iostream>
2  using namespace std;
3  int main() {
4      string b; int d;
5      cout << "Adjon meg egy tízes "
6           << "számrendszerbeli számot!\n";
7      for(cin>>d; d>0; d/=2) {
8          b = char(d%2 + '0') + b;
9      }
10     cout << "Kettes számrendszerben: " << b
11          << endl;
12     return 0;
13 }
```

Szó megfordítása helyben

szofordit1.cpp

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      cout << "Adjon meg egy szot! ";
6      string szo;
7      cin >> szo;
8      int eleje, vege;
9
10     eleje = 0; vege = szo.length()-1;
11     while(eleje < vege) {
12         char csere = szo[eleje];
13         szo[eleje] = szo[vege];
14         szo[vege] = csere;
15         eleje++; vege--;
16     }
17
18     cout << "Megforditva: "
19          << szo << endl;
20     return 0;
21 }
```

szofordit2.cpp

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

int main() {
    cout << "Adjon meg egy szot! ";
    string szo;
    cin >> szo;
    int eleje, vege;

    for( eleje=0, vege=szo.length()-1;
        eleje<vege;
        eleje++, vege--) {
        char csere = szo[eleje];
        szo[eleje] = szo[vege];
        szo[vege] = csere;
    }

    cout << "Megforditva: "
         << szo << endl;
    return 0;
}
```

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
```