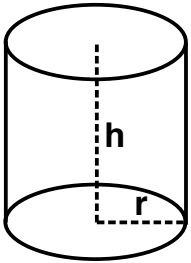
Programozás
(GKxB_INTM114)

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\talapjanak sugarat! "; cin >> r;
9    cout << "\tmagassagat! "; cin >> h;
10   cout << "A henger\n\tterfogata: " << r*r*M_PI*h
11        << "\n\tfelszine: " << 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` típussal ábrázolható legkisebb pozitív szám
  `DBL_MAX` a `double` típussal 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 $\rightarrow$ hordozhatóság
- Beszerkesztendő fejfájl: `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
`double ceil(double x)`	Az x-nél nagyobb egészek közül a legkisebbet adja
`double cos(double x)`	koszinusz
`double cosh(double x)`	hiperbolikus koszinusz
`double exp(double x)`	exponenciális fv.
`double fabs(double x)`	abszolút érték
`double fmod(double x, double y)`	osztás lebegőpontos maradékát adja
`double log(double x)`	természetes alapú logaritmus
`double log10(double x)`	10-es alapú logaritmus
`double pow(double x, double y)`	hatványozás
`double sqrt(double x)`	négyzetgyökvonás

haromszog4.cpp

```
1  #include <iostream>
2  #define OLDALSZAM 3
3  using namespace std;
4  int main() {
5    double ot[OLDALSZAM]; // racionalis szamok
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 kulon valtozoja 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`	*bármi*→`long double`
`double`	*bármi*→`double`
`float`	*bármi*→`float`
egész-előléptetés	*egész-előléptetés*
`unsigned long`	*bármi*→`unsigned long`
`long`→(`unsigned`) `long`	`unsigned int`→(`unsigned`) `long`
`long`	*bármi*→`long`
`unsigned int`	*bármi*→`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
```

fahrCels2.cpp

```
10    // Implicit tip.konv.
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
```

fahrCels4.cpp

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

Kimenet

```
Celsius: 22.2222
```

fahrCels5.cpp

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

Kimenet

```
Celsius: 0
```

kor3.cpp Ellipszis rajzolása

```
1  #include <iostream>
2  #define A 5 // Fel nagytengely (fel fotengely)
3  #define B 9 // Fel kistengely
4  using namespace std;
5
6  int main() {
7    int sor = -A;
8    while(sor <= A) {
9      int oszlop = -B;
10     while(oszlop <= B) {
11       if((sor*sor)/double(A*A) + (oszlop*oszlop)/double(B*B) <= 1) cout << '*';
12       else cout << ' ';
13       oszlop++;
14     }
15     sor++;
16     cout << '\n';
17   }
18   return 0;
19 }
```

Feltételes operátor: ?:

Szelekciós tevékenység

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

Feltételes operátor

```
valtozo = logikaiKifejezes ? ertekHaIgaz : 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 ? "Megszerkesztheto.\n" : "Ez nem szerkesztheto meg!\n");
```

abszolut1.cpp

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

abszolut2.cpp

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

Operátorok precedenciája és asszociativitása

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

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

for

```
for(ciklusValtozo=kezdoErtek; 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--) {
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  }
```

Tízes 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 tizes "
6         << "szamrendszerbeli szamot!\n";
7    cin >> d;
8    while(d > 0) {
9      c = d%2 + '0'; b = c + b; d /= 2;
10   }
11   cout << "Kettes szamrendszerben: " << b
12        << endl;
13   return 0;
14 }
```

kettes3.cpp

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