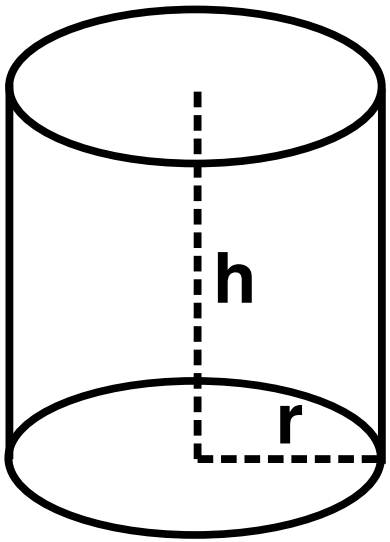
Programozás
(GKxB_INTM021)

Dr. Hatwágner F. Miklós

Széchenyi István Egyetem, Győr

2019. július 2.

Henger felszínének, térfogatának kiszámítása



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 felszínének, térfogatának kiszámítása

henger.cpp

```
#define _USE_MATH_DEFINES // regi rendszerekhez
#include <iostream>
#include <cmath> // vagy math.h
using namespace std;

int main() {
  double r, h;
  cout << "Adja meg egy henger\n\talapjanak sugarat! ";
  cin >> r;
  cout << "\tmagassagat! ";
  cin >> h;
  cout << "A henger\n\tterfogata: " << r*r*M_PI*h
       << "\n\tfelszine: " << 2.*r*M_PI*(r+h) << endl;
  return 0;
}
```

Lebegőpontos konstansok

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`

Lebegőpontos konstansok

Lebegőpontos konstansok főbb tulajdonságai, folyt.

- 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` – $\sqrt{2}$

Egész típusú konstansok, változók tulajdonságai

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)

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;
```

Egész típusú konstansok, változók tulajdonságai

Az egész konstansok főbb tulajdonságai, folyt.

- 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
```

Abszolútérték számítása

abszolut1.cpp

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

int main() {
  double v, abs;
  cout << "Szam: "; cin >> v;
  cout << "Abszolut erteke: ";


  if(v < 0.) abs = -v;
  else abs = v;


  cout << abs;
  return 0;
}
```

abszolut2.cpp

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

int main() {
  double v, abs;
  cout << "Szam: "; cin >> v;
  cout << "Abszolut erteke: ";


  abs = v<0. ? -v : v;



  cout << abs;
  return 0;
}
```

Abszolútérték számítása

Feltételes operátor: ?:

Szelekciós tevékenység

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

Feltételes operátor

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

Másodfokú egyenlet megoldása

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

```
#include <iostream>
#include <cmath> // sqrt() miatt
using namespace std;

int main() {
  double a, b, c;
  cout << "Az ax^2+bx+c = 0 egyenlet megoldasa\n";
  cout << "a erteke: "; cin >> a;
  if(a == 0.) {
    cout << "Az egyenlet nem masodfoku!\n";
  } else {
    cout << "b erteke: "; cin >> b;
    cout << "c erteke: "; cin >> c;
    double d = b*b - 4.*a*c;
    if(d < 0.) {
      cout << "Nincs valos gyok!\n";
    } else {
      cout <<   "x1 = " << (-b + sqrt(d)) / (2.*a)
           << "\nx2 = " << (-b - sqrt(d)) / (2.*a) << endl;
    }
  }
  return 0;
}
```

Matematikai függvények

- 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

Matematikai függvények

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

Háromszög megszerkeszthetőségének ellenőrzése

haromszog5.cpp

```
#include <iostream>
#define OLDALSZAM 3
using namespace std;
int main() {
  double ot[OLDALSZAM]; // nem csak egesz lehet
  int i;
  bool megszerkesztheto = false;
  char onev;
  cout << "Adja meg egy haromszog oldalhosszait!\n";
  do {
    i = 0; onev = 'A';
    while(i < OLDALSZAM) {
      do {
        cout << onev << " oldal hossza: ";
        cin >> ot[i];
      } while(ot[i] <= 0.); // lebegőpontos konstans
      i++; onev++;
    }
    megszerkesztheto = (ot[0]+ot[1]>ot[2] and
      ot[1]+ot[2]>ot[0] and ot[2]+ot[0]>ot[1]);
    // felteteles operator
    cout << (megszerkesztheto ? "Megszerkesztheto.\n" :
      "Ez nem szerkesztheto meg!\n");
  } while(not megszerkesztheto);
  return 0; }
```

Háromszög megszerkeszthetőségének ellenőrzése

haromszog6.cpp

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

Kimenet

```
Adja meg egy haromszog oldalhosszait!
65 oldal hossza:
```

Típuskonverzió

Implicit típuskonverzió: kétoperandusos műveleteknél, ha az operandusok típusa eltér. Ált. 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!

Háromszög megszerkeszthetőségének ellenőrzése

Explicit típuskonverzió (Type cast)

haromszog7.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:

Fahrenheit – Celsius átváltás

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

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 }
```

Kimenet

```
Fahrenheit -> Celsius
Fahrenheit: 72
Celsius: 0
```

Fahrenheit – Celsius átváltás

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

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

Növekményes, vagy `for` 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 = kezdoErtek;
while(ciklusValtozo < vegErtek) {
    ciklusMag;
    ciklusValtozo += lepes;
}
```

for

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

Növekményes, vagy `for` ciklus

N darab szám beolvasása, tárolása, kiírása fordított sorrendben

forditva1.cpp

```
10  int szamok[N], db;
11  db = 0;
12  while(db < N) {
13    cout << db+1 << ". szam: ";
14    cin >> szamok[db];
15    db++;
16  }
17
18
19  cout << "\nMegforditva:\n";
20  db = N-1;
21  while(db >= 0) {
22    cout << szamok[db] << '\t';
23    db--;
24  }
```

forditva3.cpp

```
10  int szamok[N], db;
11  for(db=0; db<N; db++) {
12    cout << db+1 << ". szam: ";
13    cin >> szamok[db];
14  }
15
16
17
18
19  cout << "\nMegforditva:\n";
20  for(db=N-1; db>=0; db--) {
21    cout << szamok[db] << '\t';
22  }
```

Növekményes, vagy `for` ciklus

Általános forgatókönyv:

while

```
utasitas1;
while(feltetel) {
    utasitas2;
    utasitas3;
}
```

for

```
for(utasitas1; feltetel; utasitas3) {
    utasitas2;
}
```

Tízes számrendszerbeli szám átalakítása kettes számrendszerbelivé

kettes2.cpp

```
 9    cin >> d;
10    i = 0;
11    while(d > 0) {
12      b[i] = d%2+'0'; d /= 2; i++;
13    }
```

kettes3.cpp

```
 9  for(cin>>d,i=0; d>0; d/=2,i++) {
10    b[i] = d%2+'0';
11  }
```

Növekményes, vagy `for` ciklus

Szó megfordítása helyben

szofordit1.cpp

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

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


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


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

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;
}
```

Növekményes, vagy `for` ciklus

fahrCels6.cpp

```
#include <iostream>
#define ALSO 0
#define FELSO 150
#define LEPES 10
using namespace std;

int main() {
  cout << "Fahrenheit\tCelsius\n"
       << "----------\t--------\n";
  double f;


  for(f=ALSO; f<=FELSO; f+=LEPES)
    cout << f << "\t\t"
         << (5./9.)*(f-32.) << '\n';


  return 0;
}
```

Kimenet

```
Fahrenheit      Celsius
----------      --------
0               -17.7778
10              -12.2222
20              -6.66667
30              -1.11111
40              4.44444
50              10
60              15.5556
70              21.1111
80              26.6667
90              32.2222
100             37.7778
110             43.3333
120             48.8889
130             54.4444
140             60
150             65.5556
```