Programming basics
(GKNB_INTA023)

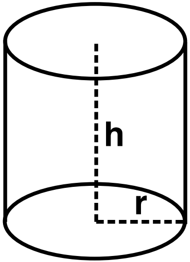
Hatwagner F. Miklós, PhD.

Széchenyi István University, Győr, Hungary

https://github.com/sze-info/ProgrammingBasics

October 7, 2020

Calculating the surface and volume of a cylinder



Tasks:

1. Read the height and radius of the cylinder
2. Calculate the surface and volume of the cylinder
   $V = r^2 \pi h$
   $S = 2r\pi h + 2r^2\pi = 2r\pi(r + h)$

Type	Size	Number representation limits	Precision
float	4 bytes	$\pm 3,4 \cdot 10^{-38}$ – $\pm 3,4 \cdot 10^{+38}$	6-7 dec. digits
double	8 bytes	$\pm 1,7 \cdot 10^{-308}$ – $\pm 1,7 \cdot 10^{+308}$	15-16 dec. digits
long double	10 bytes	$\pm 1,2 \cdot 10^{-4932}$ – $\pm 1,2 \cdot 10^{+4932}$	19 dec. digits

Calculating the surface and volume of a cylinder

cylinder.c

```c
#include <stdio.h>
#include <math.h>

int main(void) {
  double r, h;
  printf("Enter the radius of the cylinder: ");
  scanf("%lf", &r); // scanf: %lf -> double
  printf("Enter the height of the cylinder: ");
  scanf("%lf", &h); // printf: %f -> double
  printf("Volume: %f\n\tSurface: %f\n",
         r*r*M_PI*h, 2.*r*M_PI*(r+h));
  return 0;
}
```

Calculating the surface and volume of a cylinder

Main properties of floating point literals

- representation limits → `float.h`, eg.

  `DBL_MIN` the least positive normal number representable by type `double`

  `DBL_MAX` the greatest finite number that can be stored in a `double`
- the integer or the fractional part of the mantissa may be omitted, but not both of them!
- the decimal point or the exponent (`e, E`) part may be omitted, but not both of them!
- without any suffix the internal storage type is `double`

Calculating the surface and volume of a cylinder

Main properties of floating point literals, contd.

- Suffixes to change the internal storage type of a literal:
  - `f`, `F` (`float`)
  - `l`, `L` (`long double`)

Some floating point literals

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

Some of the (not necessarily standardized) literals of `math.h`

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

Calculating the surface and volume of a cylinder

Main properties of integer literals

- can be given in decimal, octal (`0...`) and hexadecimal (`0x...`, `0X...`) form
- suffixes to change the internal storage type:
  - `u`, `U` (unsigned)
  - `l`, `L` (long)

Integer variables and literals

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

Calculating the surface and volume of a cylinder

Main properties of integer literals, contd.

- representation limits of platform-dependent integer types → `limits.h`
- platform-independent, fixed size integer types, eg. `int32_t`, `uint16_t` → `stdint.h` (C99).

some details of `limits.h`

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

Calculating absolute value

absolute1.c

```c
#include <stdio.h>

int main(void) {
  double v, abs;
  printf("Number: ");
  scanf("%lf", &v);
  printf("Absolute value: ");


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


  printf("%f\n", abs);
  return 0;
}
```

absolute2.c

```c
#include <stdio.h>

int main(void) {
  double v, abs;
  printf("Number: ");
  scanf("%lf", &v);
  printf("Absolute value: ");


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



  printf("%f\n", abs);
  return 0;
}
```

Calculating absolute value

Ternary, conditional operator (shorthand for if...else): ?:

if ... else

```
if(logicalExpression) {
    variable = valueIfTrue;
} else {
    variable = valueIfFalse;
}
```

Ternary operator

```
variable = logicalExpression ? valueIfTrue : valueIfFalse;
```

Triangle inequality

triangle5.c

```c
#include <stdio.h>
#include <stdbool.h>
#include <iso646.h>
#define SIDES 3

int main(void) {
  double sideArray[SIDES]; // side lengths can be racional numbers, too
  int i;
  bool valid = false;
  printf("Enter the sides of a triangle!\n");
  do {
    i = 0;
    while(i < SIDES) {
      do {
        printf("Length of side %c: ", 'A'+i);
        scanf("%lf", &sideArray[i]);
      } while(sideArray[i] <= 0.); // floating-point literal
      i++;
    }
    valid = (sideArray[0]+sideArray[1]>sideArray[2] and
             sideArray[1]+sideArray[2]>sideArray[0] and
             sideArray[2]+sideArray[0]>sideArray[1]);
    // ternary operator
    printf("The triangle is %s.\n", (valid ? "valid" : "invalid"));
  } while(not valid);
  return 0; }
```

Solving a quadratic equation

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

```c
#include <stdio.h>
#include <math.h> // sqrt() needs it

int main(void) {
  double a, b, c;
  printf("Solving equation ax^2+bx+c = 0\n"
         "Enter the value of coefficient a: ");
  scanf("%lf", &a);
  if(a == 0.) {
    printf("The equation is not quadratic!\n");
  } else {
    printf("Enter the value of coefficient b: "); scanf("%lf", &b);
    printf("Enter the value of coefficient c: "); scanf("%lf", &c);
    double d = b*b - 4.*a*c;
    if(d < 0.) {
      printf("The equation has no real root.\n");
    } else {
      printf("x1 = %f\nx2 = %f\n", (-b + sqrt(d)) / (2.*a),
                                   (-b - sqrt(d)) / (2.*a));
    }
  }
  return 0;
}
```

Solving a quadratic equation

Mathematical functions

- Standard function libraries → portability
- Header to be included: `math.h`
- GCC: linking of the floating-point library must be explicitely stated, eg.:

  ```
  gcc -Wall -o quadratic quadratic.c -lm
  ```
- The type of function parameters and return values are usually `double`
- Argument and return value of trigonometric functions are specified in radians

Solving a quadratic equation

Some often used mathematical function

Prototype	Goal
`double ceil(double x)`	returns the smallest integral value that is not less than x
`double cos(double x)`	cosine
`double cosh(double x)`	hyperbolic cosine
`double exp(double x)`	base-e exponential function
`double fabs(double x)`	absolute value of floating-point number
`double fmod(double x, double y)`	computes the floating-point remainder of dividing x by y
`double log(double x)`	natural logarithmic function
`double log10(double x)`	base-10 logarithmic function
`double pow(double x, double y)`	power function
`double sqrt(double x)`	square root

Fahrenheit – Celsius conversion

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

fahrCels1.c

```c
#include <stdio.h>

int main(void) {
  printf("Fahrenheit --> Celsius\n"
         "Fahrenheit: ");
  double f;
  scanf("%lf", &f);
  // Integer division, implicit type conversion
  printf("Celsius: %f\n", (5/9)*(f-32));
  return 0;
}
```

Output

```
Fahrenheit -> Celsius
Fahrenheit: 72
Celsius: 0.000000
```

Remarks:

- 5/9 → always 0!
- `f-32` → implicit type conversion to `double`