

## Computer Programming: Skills & Concepts (CP)

### Arithmetic operations, int, float, double

C. Alexandru

26 September 2017

CP Lect 4 – slide 1 – 26 September 2017

## Today's lecture

- ▶ Arithmetic Operations for int
- ▶ Quadratic Equations.
- ▶ More types: double (and float).

CP Lect 4 – slide 3 – 26 September 2017

## Monday's lecture

- ▶ Variables and change-of-state
- ▶ The “squaring” problem.
- ▶ *Types* of variables: int.
- ▶ Assigning and re-assigning values to a variable.
- ▶ The if-statement.
- ▶ Input using scanf.

CP Lect 4 – slide 2 – 26 September 2017

## Arithmetic Operators for int

- + Addition.
- Subtraction *or* negation.
- \* Multiplication (don't use x).
- / Division – order is important here!
  - ▶ What is 4/2 ?
  - ▶ What is 5/2 ?
- % Integer remainder (eg, 5 % 3 = 2).
  - ▶ You've seen % used for something else ...
  - ▶ nothing whatsoever to do with this % !
- ++ Increment (x++ means x = x+1).
- Decrement (x-- means x = x-1).

^ (sometimes used in 'real life' for powers – e.g., x<sup>3</sup>) is NOT an arithmetic operation in the *C programming language* – for powers, use the \* operator (repeatedly) or the pow function from math.h.

CP Lect 4 – slide 4 – 26 September 2017

## Solving quadratic equations

Consider any quadratic polynomial of the form  $ax^2 + bx + c$ ,  $a \neq 0$ . We know this equation has exactly two *complex* roots (solutions to  $ax^2 + bx + c = 0$ ) given by:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

**Suppose we want real roots ONLY.**

Three cases:

- ▶ If  $b^2 < 4ac$ , there are **no** real solutions.
- ▶ If  $b^2 = 4ac$ , there is **one** (repeated) real solution:  $-b/(2a)$ .
- ▶ If  $b^2 > 4ac$ , there are **two** different real solutions.

CP Lect 4 – slide 5 – 26 September 2017

## Real numbers in C

For working with “real numbers” in C, there are two standard options: `float` and `double`. Neither type can truly represent all real numbers – both types have a limited number of significant digits. But they work well as an approximation for reals.

We will require the coefficients input for the quadratic equation to be `int`. However we will also need some `float` or `double` variables for the roots.

CP Lect 4 – slide 7 – 26 September 2017

## C program to Solve Quadratic Equations

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

Steps of our program:

- ▶ Take in the inputs  $a$ ,  $b$  and  $c$  from the user (`scanf`).
- ▶ *check that  $b^2 - 4ac$  is non-negative.*
  - ▶ If negative, output a message about “No real roots”.
  - ▶ If positive, proceed.
- ▶ Get the square root of  $b^2 - 4ac$ .
- ▶ Output both roots (or one if repeated).
- ▶ `return EXIT_SUCCESS;`

*We cannot continue working with `int` variables only.  
We do not expect the roots to be integers even when  $a, b, c$  are.*

CP Lect 4 – slide 6 – 26 September 2017

## Types: float

- ▶ A signed floating-point number: *numbers with decimal points.*
- ▶ Form to write a float is a decimal number optionally followed by `e` (or `E`) and an integer *exponent*:
- ▶ For example:
  - ▶ 1.5, -2.337, 6e23 (*having values 1.5, -2.337 and  $6 \times 10^{23}$* )
  - ▶ 0.0, 0., .0 (*all of these have value 0.0*)
- ▶ Accurate to about 7 significant digits:
  - ▶ Max value is  $3.402823 \times 10^{38}$  on DICE (system dependent);
  - ▶ Requires the same amount of storage as `int`.
- ▶ Contrast with real numbers in mathematics?
- ▶ `printf("%f", floatVar)` and `scanf("%f", &floatVar)`.
  - ▶ `%f` means “float”
- ▶ Stored in 32-bit sign(1)/exponent(8)/mantissa(23) representation.

CP Lect 4 – slide 8 – 26 September 2017

## Types: double

- ▶ A float with double precision.
- ▶ Same form for writing double as float in programs.
- ▶ Accurate to about 15 significant digits:
  - ▶ Max value is  $1.7976931348623157 \times 10^{308}$ ;
  - ▶ Requires twice the storage space of float;
  - ▶ Values may depend on your computer.
- ▶ `printf("%lf", doubleVar)` and `scanf("%lf", &doubleVar)`
  - ▶ The `%lf` means 'long float'.
  - ▶ Actually, the C standard says you should `printf("%f",doubleVar)`; but most compilers also allow `%lf`, which is more consistent. Use either, but
    - ▶ remember you **must** use `"%lf"` to scan a double.
- ▶ Stored in 64-bit sign(1)/exponent(11)/mantissa(52) representation.

CP Lect 4 – slide 9 – 26 September 2017

## Writing float/double in programs

```
#include <stdlib.h>
#include <stdio.h>

int main(void) {
    float x, x2;
    double y, y2;
    x = 1e8 + 5e-4;
    x2 = -0.2223;
    y = 1e8 + 5e-4;
    y2 = -6e306;
    printf("Two floats are %f\n and %f.\n", x, x2);
    printf("Two doubles are %lf\n and %lf.\n", y, y2);
    return EXIT_SUCCESS;
}
```

CP Lect 4 – slide 11 – 26 September 2017

## float or double ?

- ▶ floats are not precise enough for most scientific or engineering calculations, so
- ▶ the standard maths libraries all work with doubles, so
- ▶ always use doubles unless you have a good reason to use floats
- ▶ (for example, if you're doing *lots* of computation on *lots* of numbers; or in some graphics applications where double precision is useless)
- ▶ and anyway, 9.36 is *really* a double – to get an actual float, you have to write 9.36f

CP Lect 4 – slide 10 – 26 September 2017

## Output from float/double

```
zagreb: ./a.out
Two floats are 100000000.000000
and -0.222300.
Two doubles are 100000000.000500
and -60000000000000000415146435945218699544294763362085459
8420126115503945248872404569187418808157783928463113189413
9451804157162361475827507299487506852076765339123136457002
1480187142842148415306933169404320733422827669951287867963
4094905773013933547655429167101887147924700636668768497796
83791229808236015124480.000000.
```

Is there a mistake in the printing out of x and of y2?

**No! The first few digits are correct** (float (resp. double) guarantees the first 7 (resp. 15)).

CP Lect 4 – slide 12 – 26 September 2017

## double vs float – example

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    double x = 0.0;
    int i = 0;
    while ( i < 1000000 ) {
        x = x + 0.9; i = i + 1;
    }
    printf("%f\n",x);
    return EXIT_SUCCESS;
}

prints:
900000.000015
```

```
#include <stdio.h>
#include <stdlib.h>
int main() {
    float x = 0.0;
    int i = 0;
    while ( i < 1000000 ) {
        x = x + 0.9; i = i + 1;
    }
    printf("%f\n",x);
    return EXIT_SUCCESS;
}

prints:
892043.562500
an error of almost 1% !
```

CP Lect 4 – slide 13 – 26 September 2017

## Reading material

Sections 2.8, 2.9, 2.10, 2.11 of “A book on C” discuss **Operators, Operator precedence, and assignments** (ie, material from Monday’s lecture).

Section 3.6 (The Floating Types) of “A Book on C”.

CP Lect 4 – slide 15 – 26 September 2017

## Mixing Types, and Casting

- ▶ / does *integer division* on **ints**:  $3/2 \rightarrow 1$
- ▶ It does real division on **doubles**:  $3.0/2.0 \rightarrow 1.5$ .
- ▶ What if we mix **doubles** and **ints**?  $3.0/2 \rightarrow ?$   $3/2.0 \rightarrow ?$
- ▶ The **int** gets *promoted* to **double**:  
 $3.0/2 \rightarrow 3.0/2.0 \rightarrow 1.5$  and  $3/2.0 \rightarrow 3.0/2.0 \rightarrow 1.5$
- ▶ This happens with all arithmetic operators. **BUT** beware that it happens ‘from the inside out’:  
 $(5/2)*1.2 \rightarrow 2*1.2 \rightarrow 2.4$
- ▶ If **int**  $x,y$ ; how do we do real division of  $x$  by  $y$ ?
- ▶ Can use promotion:  $(x*1.0)/y \rightarrow x^{dbl}/y \rightarrow x^{dbl}/y^{dbl}$
- ▶ Clearer and safer to **cast**: explicitly convert types:  
 $(double)x/(double)y \rightarrow x^{dbl}/y^{dbl}$
- ▶ Be careful:  $(double)(5/2) \rightarrow (double)(2) \rightarrow 2.0$
- ▶ Alternatively:  
`double xd, yd;`  
`xd = x; yd = y; xd/yd`

CP Lect 4 – slide 14 – 26 September 2017