Lecture 3

We know types!

- int,
- float, double
- char,
- bool
- void
- example ...

Variables

Substitution and assignment:

```
int a,b; //declare two variables of type int
a=35; //a store value of 35
b=6; //b store 6
a=a+b;// perform addition in temporary space, copy on to a
```

An arithmetic expresion:

```
double x1; //declare a variable of type double
x1=(-b+sqrt(delta))/(2.0*a); //Perform RHS operation, write result
    to x1
```

Example ...

Substitution

When using "=" sign variable on the LHS is assigned value of RHS. This does not necessarily means equality!

The RHS is calculated first and later the value is copied to the LHS.

Types of LHS and RHS should be the same.

Mixing of types should be avoided.

```
double x1=6.28;
int a = 2
a = x1; //loss of data since a=6!
x1=a;
```

There is an explicit way to change the type: casting

```
double x1=6.28;
int a = 2
a = (int)x1; //loss of data, but no warning
x1=(double)2/3; //x1 is not zero
```

Precedence of operators

The ones we know so far

1. () brackets
2. + - unary plus/minus: (-1)
3. * / % binary operator a*b
4. - + binary operator a+b

```
-5 * 3 + 4 * 5. / 2.
((-5)*3)+(4*5)/2.
```

examples ...

Increment/decrement operators

Increment / decrement operators are unary operators that change the value of a variable by 1.
They can have postfix or prefix form

```
a++ //postfix
a--
++a //prefix
--a
```

```
int a = 1;
int b = a++; // stores 1+a (which is 2) to a
             // returns the value of a (which is 1)
             // After this line, b == 1 and a == 2
a = 1;
int c = ++a; // stores 1+a (which is 2) to a
             // returns 1+a (which is 2)
             // after this line, c == 2 and a == 2
```

example ...

Compound Assignment Operators += -= *= /=

Change the value of the RHS by the value of the LHS

```
a+=b; // same as a = a +b;
a-=b; // same as a = a -b;
a*=b; // same as a = a *b;
a/=b; // same as a = a /b;
```

```
int a = 1;
int b = 5;
a+=b;
a-=1;
a*=9;
a/=b;
...
```

example ...

printf()

```
int a = 1;
float b=4.78;
double c=3.4e9;
char d='e'
printf("%d␣%f␣%e␣%c␣\n", a, b, c,
     d)
```

- *#include <stdio.h>*
- Sends formatted output to stdout - the screen
- *int printf(const char *f, ...)* returns the total number of characters written
- *f*-text to be written, might contain format tags, replaced by value provided in arguments. Format tag is as follows: **%specifier**
- We know some specifiers: c, d or i, e, E, f, g, o, u, x or X
- examples ...

new stuff

```
int a = 1;
float b=4.78;
double c=3.4e9;
char d='e'
printf("%+d␣%-f␣%10e␣%c␣\n", a, b,
      c, d)
printf("%.10d␣%10.10f␣%10e␣%c␣\n",
      a, b, c, d)
```

- Format tag might contain more information
- **%[flags][width][.prec]spc**
- Some Flags: - left justify, +force sign
- width - minimum number of characters to be printed
- .prec - for ints: the minimum number of digits to be written
  For e, E and f: the number of digits to be printed after the decimal point.
  For g and G: This is the maximum number of significant digits to be printed.
- examples ...

The cursor control

```
printf("␣aaa␣b\b␣\n␣fff␣\t␣ff␣\v␣
    fff␣\r␣a␣")
```

- \b - backspace
- \n - carriage return and newline
- \t - tab
- \v - newline
- \r - carriage return
- \\- \
- \" - '
- examples ...

scanf()

```
int a;
scanf("%d", &a);//Read int
```

He forgot abouth &

Source: the XKCD commic

- *#include <stdio.h>*
- Reads formatted input from stdin - the keybord
- *int scanf(const char *f, ...)* returns the total number of characters written
- *f*- a string containing a format specifier **%specifier**
- **Mind the &** - very very important!!
- examples ...

math.h
trigonometric

- Defines various mathematical functions
- examples ...

```
#include <math.h>
double acos(double x) Returns the arc cosine of x in radians.
double asin(double x) Returns the arc sine of x in radians.
double atan(double x) Returns the arc tangent of x in radians.
double atan2(doubly y, double x)Returns the arc tangent in radians
    of y/x based on the signs of both values to determine the
    correct quadrant.

double cos(double x) Returns the cosine of a radian angle x.
double cosh(double x) Returns the hyperbolic cosine of x.
double sin(double x) Returns the sine of a radian angle x.
double sinh(double x) Returns the hyperbolic sine of x.
double tanh(double x) Returns the hyperbolic tangent of x.
```

math.h

```
#include <math.h>
double exp(double x) Returns the value of e raised to the xth power.
double log(double x)Returns the natural logarithm (base-e logarithm)
      of x.
double log10(double x)Returns the common logarithm (base-10
     logarithm) of x.
double pow(double x, double y)Returns x raised to the power of y.
double sqrt(double x)Returns the square root of x.
double ceil(double x)Returns the smallest integer value greater than
      or equal to x.
double fabs(double x)Returns the absolute value of x.
double floor(double x)Returns the largest integer value less than or
      equal to x.
```

for e^x use *exp(x)*, never use *pow(exp(1), x)* ...
Examples!