Data Structures in C

Prof. Georg Feil

Dangers of Global and Uninitialized Variables

Summer 2018

Global Variables

- Recall that variables declared outside of functions (without the keyword 'static') are global
- Typically, global variable declarations go near the top of the source file

```
#include <stdio.h>
double g_radius = 7.8;  // Radius in centimeters

int main(int argc, char** argv)
{
   …
   printf("The radius is %f\n", g_radius);
}
```

Dangers of global variables

- Global variables can be accessed or changed by any part of your program at any time
  - … a bit like making every field variable public in Java
  - Leads to poor program structure, hard-to-find bugs
  - *breaks encapsulation*

- You should use global variables only when you absolutely really need to… otherwise avoid using global variables
  - If it can be a local variable, make it local
  - If you can pass a value as a function parameter, do so
  - If it can't be local but is only used in that source file, make it a module variable using 'static'
  - … Do this now to your program from Exercise 1 in the "Variables" slides

Dangers of uninitialized variables

- In C you can accidentally use uninitialized local variables
  - The compiler will not give you an error
  - Our compiler gcc can give you a warning, but only if -Wall is specified *and* may also need the -O option to turn the optimizer on

- What is the value of an uninitialized local variable in C?
  - Undefined! Unpredictable!
  - Literally whatever values were in that part of memory before, could be different every run of the program, or only every millionth run

- What is the value of an uninitialized global/module variable in C?
  - Zero (false for Boolean, null for pointers)

Exercise: Uninitialized Variables

- Write a C program that declares an int variable (without initializing), then prints it out
- Try it with a local variable and a global variable

- Hint: It may not be so easy to "see" the problem with an int
  - Try using a long long int, and print using `%lld`
  - Try using a double, and print using `%g`
  - Try adding some other local variables, e.g. one or two doubles