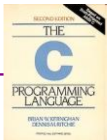
C

David Notkin ● Autumn 2009 ● CSE303 Lecture 8

Lecture summary

- History and characteristics of C
- Major C language features
  - differences between C and Java
- basic console input and output (printf and scanf)
- Our learning objectives in C
  - procedural programming
  - deeper understanding of program compilation and execution
  - learn details of memory management
  - debugging skills
  - software development strategies

History



- Created in 1972 by Dennis Ritchie of Bell Labs to accompany the Unix operating system
  - latest version standard: "C99" (1999)
- Designed for creating system software (programs close to the OS that talk directly to hardware)
  - Also designed to be hardware-independent (portable)
  - C is also used to develop high-level applications
- Currently one of the top two most widely used language worldwide
- Based on ALGOL; has influenced the designs of many languages
  - C++, Java, C#, Perl, Eiffel, Objective-C, Modula, Pascal, ...

Characteristics of C

- fairly similar basic syntax and semantics to Java
  - **`if/else`**, **`for`**, **`while`**, **`int`**, **`double`**, **`{}`** **`[]`** **`()`** **`;`** **`+-`** **`*/%`** **`++`**
- Much smaller provided standard library than Java
- More low-level (more work for programmer, less for compiler)
- Procedural (not object-oriented)
  - C does not have objects as we know them
  - **`verb(noun);`** rather than **`noun.verb();`**
- More unsafe (an incorrect program can cause more damage): C programs have more direct access to the system / hardware

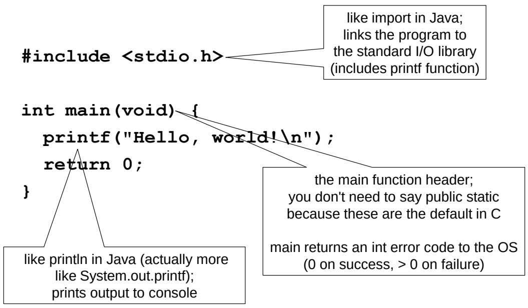
First C program

```
#include <stdio.h>

int main(void) {
  printf("Hello, world!\n");
  return 0;
}
```

- Kernighan and Ritchie started the convention that the first program you show in a new language should be one that prints "Hello, world!"

Dissecting Hello World



```
#include <stdio.h>

int main(void) {
  printf("Hello, world!\n");
  return 0;
}
```

like import in Java;
links the program to
the standard I/O library
(includes printf function)

the main function header;
you don't need to say public static
because these are the default in C

main returns an int error code to the OS
(0 on success, > 0 on failure)

like println in Java (actually more
like System.out.printf);
prints output to console

Second C program

```
/* Computes greatest common divisor (GCD) with Euclid's algorithm. */
#include <stdio.h>

int main(int argc, char** argv) {
  int a, b, temp, r;

  printf("Please enter two positive integers: ");
  scanf("%d %d", &a, &b);

  if (b > a) {
    temp = a;
    a = b;
    b = temp;
  }

  while ((r = a % b) != 0) {
    a = b;
    b = r;
  }

  printf("The GCD is %d.\n", b);
  return 0;
}
```

Compiling/running

command	description
`gcc`	GNU C compiler

- To compile a program
  - `gcc -o target source.c`
  - `target` is the name of the executable program to build
- The compiler builds an actual executable file, not a `.class` like Java
  - example: `gcc -o hi hello.c`
- To run your program, just execute that file
  - example: `./hi`

gcc options (partial)

option	description
`-W`	level of warnings to display (common usage: `-Wall` for all warnings)
`-o`	output executable file name (if omitted, compiles to file `a.out`)
`-g`	generates information for debugger tools

- Most common usage for this course:
  - `gcc -g -Wall -o target source.c`
  - the warnings from `-Wall` will protect us from unwise idioms

Guess: how many pages does the gcc manual have on gcc options?

-Waddress
-Warray-bounds (only with '-O2')
-Wc++0x-compat
-Wchar-subscripts
-Wimplicit-int
-Wimplicit-function-declaration
-Wcomment
-Wformat
/ObjC and unless '-ffreestanding')
-Wmissing-braces
-Wnonnull
-Wparentheses
-Wpointer-sign
-Wreorder
-Wreturn-type
-Wsequence-point
-Wsign-compare (only in C++)
-Wstrict-aliasing
-Wstrict-overflow=1
-Wswitch
-Wtrigraphs
-Wuninitialized
-Wunknown-pragmas
-Wunused-function
-Wunused-label
-Wunused-value
-Wunused-variable
-Wvolatile-register-var

printf

function	description
`printf`	prints formatted output to `stdout`

- `printf("format string", parameters);`
- A format string contains placeholders to insert parameters into it:
  - `%d` or `%i` an integer
  - `%lf` a double ('long floating-point')
  - `%s` a string
  - `%p` a pointer (seen later)

```
int x = 3; int y = 2;
printf("(%d, %d)\n", x, y);    // (3, 2)
```

printf continued

- A placeholder can specify the parameter's width or precision:
  - `%8d` an integer, 8 characters wide, right-aligned
  - `%-8d` an integer, 8 characters wide, left-aligned
  - `%.4f` a real number, 4 digits after decimal
  - `%6.2f` a real number, 6 total characters wide, 2 after decimal
- Examples:

```
int age = 45;
double gpa = 1.2345678;
printf("%8d %7.3f\n", age, gpa);
printf("%8.2f %.1f %10.5f", gpa, gpa, gpa);
```

Very much the same as Java

- General syntax for statements, control structures, function calls
- Types `int, double, char, long`
  - type-casting syntax
- Expressions, operators, precedence

```
+ - * / % ++ --
= += -= *= /= %=
< <= == != > >= && || !
```

- Scope (within set of `{ }` braces)
- Comments: `/* ... */, //` (`//` not officially legal until C99)

Mostly the same as Java

- Variables
  - can be used without being initialized (!)
  - must be declared at the start of a function or block (changed in C99)
- `for` loops
  - variable cannot be declared in the loop header
- `if/else` statements, `while` and `do/while` loops
  - there is no `boolean` type (changed in C99)
  - any type of value can be used as a test
  - 0 means false, every other number means true
- Parameters / returns
  - C has certain features for values vs. references ("pointers")

Very different from Java

- Strings
  - very clunky to use in C; arrays of characters
  - are not objects; do not contain methods (external string functions)
- I/O to/from console and files
  - no Scanner; must use input functions such as scanf
  - console I/O different than file I/O
- Errors and exceptions
  - C has no try/catch and does not represent errors as objects
  - errors are usually returned as integer error codes from functions
  - crashes are mostly called "segmentation faults" and are not of much direct utility in figuring out what is wrong

Also very different

- Arrays
  - are just bare contiguous blocks of memory
  - have no methods and do not know their own length (!)
- Objects
  - C doesn't have them
  - closest similar feature: struct (a set of fields; no methods)
- Memory management
  - most memory that you consume, you must explicitly free afterward
- API and provided libraries
  - C doesn't have very many, compared to Java
  - you must write many things yourself (even data structures)

scanf

function	description
`scanf`	reads formatted input from console

- `scanf("format string", variables);`
- uses same syntax for formatted strings, placeholders as `printf`
- Must precede each variable with an `&` (address-of operator)

```
int x;
int y;
printf("Type your x and y values: ");
scanf("%d %d", &x, &y);
```

scanf continued

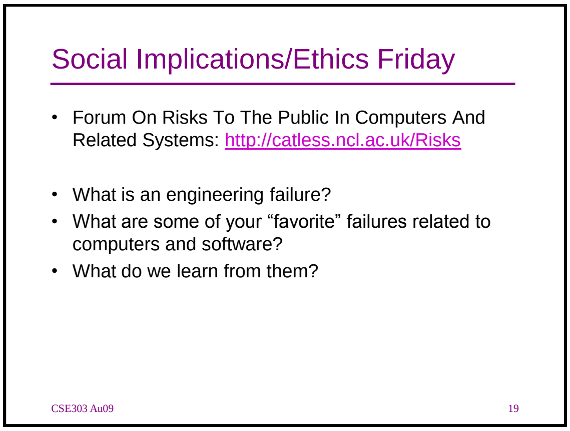
- `scanf` returns the number of values successfully read: can be examined to see whether the reading was successful
- if # of variables listed doesn't match # of format placeholders
  - too many variables: later ones ignored
  - too few variables: program crashes!
- Can match a specific input pattern

```
int x;
int y;
printf("What is your (x, y) point?\n");
scanf("My point is (%d, %d)", &x, &y);
```

Practice exercise

- Write a C program that makes change:
  - prompts the user for an amount of money
  - reports the number of pennies, nickels, dimes, quarters, and dollars
- Example

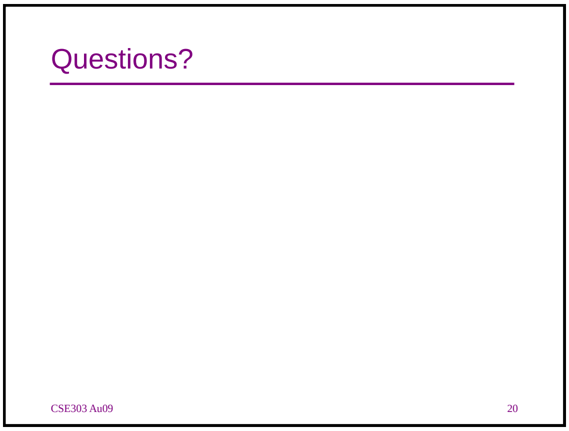
```
Amount of money? 17.93
Pennies :   2
Nickels :   1
Dimes   :   1
Quarters:   3
Dollars :  17
```



Social Implications/Ethics Friday

- Forum On Risks To The Public In Computers And Related Systems: http://catless.ncl.ac.uk/Risks

- What is an engineering failure?
- What are some of your “favorite” failures related to computers and software?
- What do we learn from them?



Questions?