

Beginning C Programming for Engineers

R. Lindsay Todd

Rensselaer Polytechnic Institute

Review

Outline

- 1 Concepts
- 2 Types
- 3 Conditions and Iteration
- 4 Functions
- 5 Arrays and Bit Operations
- 6 Uncovered Topics

Programming Basics

- *Understand the problem* you are trying to solve.
- *Design an algorithm* for your program. *Flowcharts* or *pseudocode* may be helpful.
- Solve your problem in larger *modules*, then *refine* those modules.
- After you have *edited* and *compiled* your program, check that the results make sense.

Debug

Design

Edit: `xemacs`Compile and link:
`gcc`

- 1 Preprocessor
- 2 Compiler
- 3 Linker

Execute: `./a.exe`

- 1 Loader
- 2 *Your program!*

Types, input, output

- An object's *type* indicates how data in the object can be interpreted.
- C has various types:

int	integer
float	floating point
char	character
short or short int	short integer
long or long int	long integer
double	long floating point
long double	very long floating point

- When a *variable* is declared, its type must be given.

- *Output* is result data coming from a program. *Input* is data provided to a program.
- In C, we use *printf* for output, and *scanf* for input, using these *format codes*:

```
%d, %i      integer
%f, %e, %g  float
%s          string
```

- Don't forget the `\n` in `printf`, or the `&` in `scanf`.

Conditions and Loops

- Normally program instructions are executed sequentially.
- *Conditional execution* is choosing a set of instructions based on a *condition*.
- *Iteration* or *looping* is repeating instructions in a controlled manner.
- Many loops have a well-defined *control variable*.

Syntax of if

```
/* Previous... */  
if (expression) statement  
else eStatement  
/* Following... */
```

Syntax of while

```
while (expression) statement
```

Syntax of for

```
for (initialize ; testExpr ; step) statement
```

Functions

- A *function* is a subprogram that takes parameters, returns results, and may have *side effects*.

Syntax of function definition

```
returnType name(parameters...) {  
    statements...  
}
```

Arrays and Bit Operations

- An *array* simulates a subscripted variable.
- In C, an array is a variable with one or more *bounds* declared in square brackets:

```
int a[5], b[10];
```

- Various *bitwise operators* work on the bit patterns of values:

		AND	OR	XOR
x	y	x&y	x y	x^y
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

x	~x
0	1
1	0

Uncovered Topics

- Pointers
- Custom types, `struct`, `union`, and bit fields
- Data structures
- Object-oriented programming
- Databases and file I/O
- Graphical user interfaces
- Computational complexity
- Lots of other things...