Computer Programming: Skills & Concepts
(INF-1-CP)
Practical Programming

Julian Bradfield

Tuesday 18 October 2016
This Lecture

- writing a program from scratch
- basic debugging with printf

The Task

Write a program which requests two (decimal) integers \( n \) and \( b \) from the user, and prints the representation of \( n \) in base \( b \).
Numbers in different bases

Our usual way of writing numbers is decimal or base 10:
345 means $3 \times 10^2 + 4 \times 10^1 + 5 \times 1$. 

It is sometimes convenient to use other bases. In computing, we often use base 16 (hexadecimal), with digits 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F. For example:

$1C7_{16} = 1 \times 16^2 + 12 \times 16^1 + 7 = 455$. 

Internally, computers use base 2 (binary):

$100111_2 = 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 = 39$. 

Some bases used by other cultures include 2, 5, 10, 12, 20, 60. (And of course traditional measurements use mixed and fractional bases: 12 inches to a foot, 3 feet to a yard, 5 $\frac{1}{2}$ yards to a rod, 4 rods to a chain, 10 chains to a furlong, 8 furlongs to a mile. Let's not go there.)
Numbers in different bases

Our usual way of writing numbers is decimal or \textit{base 10}: 345 means $3 \times 10^2 + 4 \times 10^1 + 5 \times 1$.

It is sometimes convenient to use other bases. In computing, we often use base 16 (hexadecimal), with digits 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F. For example: $1C7_{16} = 1 \times 16^2 + 12 \times 16^1 + 7 = 455$. 
Numbers in different bases

Our usual way of writing numbers is decimal or *base 10*:
345 means $3 \times 10^2 + 4 \times 10^1 + 5 \times 1$.

It is sometimes convenient to use other bases. In computing, we often use base 16 (hexadecimal), with digits 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F. For example:

$1C7_{16} = 1 \times 16^2 + 12 \times 16^1 + 7 = 455$.

Internally, computers use base 2 (binary):

$100111_2 = 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 = 39$
Numbers in different bases

Our usual way of writing numbers is decimal or *base 10*: $345$ means $3 \times 10^2 + 4 \times 10^1 + 5 \times 1$.

It is sometimes convenient to use other bases. In computing, we often use base 16 (hexadecimal), with digits 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F. For example:

$1C7_{16} = 1 \times 16^2 + 12 \times 16^1 + 7 = 455$.

Internally, computers use base 2 (binary):

$100111_2 = 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 = 39$

Some bases used by other cultures include 2, 5, 10, 12, 20, 60.
Numbers in different bases

Our usual way of writing numbers is decimal or \textbf{base 10}: 345 means $3 \times 10^2 + 4 \times 10^1 + 5 \times 1$.

It is sometimes convenient to use other bases. In computing, we often use base 16 (hexadecimal), with digits 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F. For example:

$1C7_{16} = 1 \times 16^2 + 12 \times 16^1 + 7 = 455$.

Internally, computers use base 2 (binary):

$100111_2 = 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 = 39$.

Some bases used by other cultures include 2, 5, 10, 12, 20, 60. (And of course traditional measurements use mixed and fractional bases: 12 inches to a foot, 3 feet to a yard, 5$\frac{1}{2}$ yards to a rod, 4 rods to a chain, 10 chains to a furlong, 8 furlongs to a mile. Let’s not go there.)
First Step

Stop!

Think!

Is the task specification complete? If not, what decisions do we need to make?
Plan

▶ Set up skeleton program;
▶ develop program incrementally;
▶ at each stage, insert debugging information;
▶ at each stage, test.
Setting Up

For trivial programs like this, can just type from nothing. In larger settings, will often copy pre-existing template (as done in labs).

Skeleton Program

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

int main(void) {

    return EXIT_SUCCESS;
}
```
And on with the job
Tips to remember

Check the program compiles after every change – and keep changes small. Test functionality whenever you can. Write test functions if possible. If you don’t understand what your program is doing, add `printf`s and trace what’s happening to your variables. (Advanced: use a `debugger` – but they have a steep learning curve.)

*Edit–compile–run* should be thought of as *edit–compile–test.*