

Simple Programs in C

CSE 130: Introduction to C Programming
Spring 2005

1

Programming Process

1. Specify the problem to be solved
2. Develop an appropriate algorithm
3. Code the algorithm using a programming language (i.e., C)
4. Test the resulting code

2

Why C?

- C is a programming “lingua franca”
- Millions of lines of C code exist
- Many other languages use C-like syntax
- C is “portable”
- C compilers exist for most platforms
- C is used for embedded systems, operating systems, and thousands of applications

3

Programming Languages

- Programming language: a form of notation used to describe an algorithm to a computer
- As programmers, we are concerned with:
 - *syntax* = the rules of the language
 - *semantics* = the meaning of a program
- The compiler will only check syntax for you!

4

The Evolution of Programming Languages

- Generation 1: binary/machine language
 - 1's and 0's
- Generation 2: assembly language
 - slightly more human-readable
- Generation 3: high-level languages
 - includes C, C++, Java, etc.

5

Types of Errors

1. Syntax Errors
 - ◁ Incorrect program “grammar”
2. Run-Time Errors
 - ◁ Illegal operation during execution
 - ◁ Ex. Division by zero
3. Logic Errors
 - ◁ Incorrect program results

6

Creating a Program

- Edit-Compile-Execute cycle:
 1. The source code is edited (written or modified) by the programmer
 2. The source code is compiled (transformed) into a machine-readable form
 3. The resulting program is executed and tested

7

Compiling A Program

- gcc — the GNU C Compiler
 - gcc is installed on Sparky
 - Usage:
 - sparky% gcc *filename.c*
 - sparky% gcc -o *name filename.c*
- Other compilers include Borland C++ and Microsoft Visual Studio

8

Executing A Program

- To execute a compiled program on Sparky, type *./filename*
- For example:
 - sparky% *./a.out*
 - sparky% *./myprog*

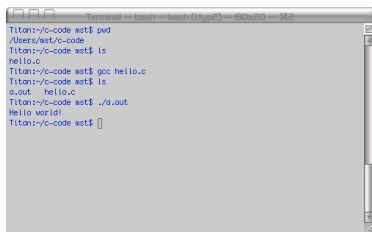
9

A First Program

```
/* My first C program */
#include <stdio.h>
int main (void)
{
    printf("Hello world!\n");
    return 0;
}
```

10

Program Compilation



```
Titoni~/c-code$ cat pdv
/Users/tst/c-code
Titoni~/c-code$ cat is
hello.c
Titoni~/c-code$ cat $ gcc hello.c
Titoni~/c-code$ cat is
a.out hello.c
Titoni~/c-code$ cat ./a.out
hello world!
Titoni~/c-code$ cat []
```

11

Comments

```
/* My first c program */
#include <stdio.h>
int main (void)
{
    printf("Hello world!\n");
    return 0;
}
```

12

Comments on Comments

- Comments are:
 - used to document code
 - ignored by the compiler
 - delimited by `/*` and `*/`
 - required in this class
- Comments add value to your code

13

Preprocessing Directives

```
/* My first C program */
#include <stdio.h>
int main (void)
{
    printf("Hello world!\n");
    return 0;
}
```

14

#include Statements

- Our sample program uses a function (piece of code) named `printf()`
- `printf()` is defined in a file named `stdio.h`
- The `#include` statement tells the compiler that it can find the definition of `printf()` elsewhere (in `stdio.h`)
- Analogy: the bibliography of a term paper

15

Standard Libraries in C

- Standard libraries contain frequently-used functions for C programs
- Ex. input/output, math functions
- `stdio.h` is the C standard library for input and output functions
- You can also create your own libraries of common code for your programs

16

The C Preprocessor

- Files are passed to the preprocessor before they move on to the compiler
- The preprocessor:
 - strips out comments
 - makes substitutions for named constants
 - inserts the contents of `#include-d` files
- Directives to the preprocessor begin with `#`

17

The `main()` Function

```
/* My first C program */
#include <stdio.h>
int main (void)
{
    printf("Hello world!\n");
    return 0;
}
```

18

More on main ()

- Program execution begins and ends with the main () function
- Program statements are executed sequentially
- When all statements in main () have been executed, the program terminates

19

The printf Statement

```
/* My first C program */
#include <stdio.h>
int main (void)
{
    printf("Hello world!\n");
    return 0;
}
```

20

The printf Statement

- printf ():
 - sends program output to the display
 - is part of the standard I/O library
- Output is specified in a quote-enclosed "control string"
- '\n' is a special "newline" character

21

The return Statement

```
/* My first C program */
#include <stdio.h>
int main (void)
{
    printf("Hello world!\n");
    return 0;
}
```

22

Return Values

- Many functions return values
 - Ex. a mathematical function
- The main () function returns a value to the operating system to indicate program status
 - Here, 0 means "everything completed OK"

23

Program Structure

1. Preprocessor Directives
 1. #include-d files
 2. Other definitions/declarations
2. Supporting Functions
3. The main () function

24

Example Program 2

```
#include <stdio.h>
int main (void)
{
    printf("Programming is fun.\n");
    printf("Doing it in C is even more fun.\n");
    return 0;
}
```

25

Example Program 3

```
#include <stdio.h>
int main (void)
{
    printf("Testing...\n..1\n...2\n....3\n");
    return 0;
}
```

26

Variables

- Programs use variables to store data
- Variables are named blocks of memory
 - Variables must be declared before use
- Different kinds of variables store different kinds of data
 - integers, floating-point #'s, characters

27

Declaring Variables

- Variables may be declared with or without an initial value
 - `int x;`
 - `int y = 5;`
 - `int z = x;`
- Variables must be assigned a value before use

28

Variables and Memory

```
int a; /* a can hold an int value */
int b = 3; /* b holds the value 3 */
```

	a		b 3			

29

Assignments

- Assignments store values in variables
- General form:
`<target variable> = <expression> ;`
- “=” means “is assigned”, not “is equal to”!!!
- Example: `area = length * width;`

30

```
#include <stdio.h>
int main(void)
{
    int feet = 6;
    int inches = feet * 12;
    printf("%d feet = %d inches",feet,inches);
    return 0;
}
```

31

Analysis of Program 4

- feet and inches are integer variables
- %d is a placeholder for an int variable
- Program output:
6 feet = 72 inches

32

Next Time

- More on variables:
 - Data Types
 - Identifiers
 - Constants
- Getting input from the user

33