AN OVERVIEW OF C

CSE 130: Introduction to Programming in C
Stony Brook University
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
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!
TYPES OF ERRORS

➤ Syntax Errors: Incorrect program “grammar”
➤ Run-Time Errors: Illegal operations during execution
   ➤ e.g., Division by zero
➤ Logic Errors: Incorrect program results
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 Xcode and Microsoft Visual Studio
EXECUTING A PROGRAM

➤ To execute a compiled program on Sparky, type ./filename

➤ For example:

➤ sparky% ./a.out

➤ sparky% ./myprog
/* My first C program */
#include <stdio.h>

int main (void)
{
    printf("Hello world!\n");
    return 0;
}
Terminal — bash — bash (tty2) — 80x20 — %2

Titan:~/c-code mst$ pwd
/Users/mst/c-code
Titan:~/c-code mst$ ls
hello.c
Titan:~/c-code mst$ gcc hello.c
Titan:~/c-code mst$ ls
a.out hello.c
Titan:~/c-code mst$ ./a.out
Hello world!
Titan:~/c-code mst$
 Statements are terminated with a ; (semicolon)
 Groups of statements are enclosed by curly braces: { and }
 Commas separate function arguments
 Whitespace is ignored (but indentation is recommended as part of good coding style)
/* My first C program */
#include <stdio.h>
int main (void)
{
    printf("Hello world!\n");
    return 0;
}
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

- They explain how and why you are doing something
/* My first C program */
#include <stdio.h>
int main (void)
{
    printf("Hello world!\n");
    return 0;
}
#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
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
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 `#`
/* My first C program */
#include <stdio.h>
int main (void)
{
    printf("Hello world!\n");
    return 0;
}
MORE ON MAIN ()

- Program execution begins and ends with the `main()` function
- Program statements are executed sequentially
- When all of the statements in `main()` have been executed, the program terminates
/* My first C program */
#include <stdio.h>
int main (void)
{
    printf("Hello world!\n");
    return 0;
}
THE printf() STATEMENT

- The printf() function:
  - 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
/* My first C program */
#include <stdio.h>
int main (void)
{
    printf("Hello world!\n");
    return 0;
}
Many functions return values
- e.g., a mathematical function
- `main()` returns a value (exit status code) to the operating system to indicate program status
- Here, 0 means “everything completed OK”
PROGRAM STRUCTURE

1. Preprocessor Directives
   a. #include-d files
   b. Other definitions/declarations

2. Supporting Functions

3. The main() function
#include <stdio.h>

int main (void)
{
    printf("Programming is fun.\n");
    printf("Doing it in C is even more fun.\n");
    return 0;
}
EXAMPLE PROGRAM 3

```
#include <stdio.h>

int main (void)
{
    printf("Testing...
..1
...2
....3
\n");
    return 0;
}
```
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 numbers, characters
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
int a; /* a can hold an int value */

int b = 3; /* b holds the value 3 */
ASSIGNMENTS

➤ *Assignments* store values in variables

➤ General form:

\[
<\text{target variable}> = <\text{expression}>;
\]

➤ “=” means “is assigned the value”, not “is equal to”!!!

➤ Example: area = length * width;
#include <stdio.h>

int main(void)
{
    int feet = 6;
    int inches = feet * 12;
    printf("%d feet = %d inches",feet,inches);
    return 0;
}

ANALYSIS OF PROGRAM 4

➤ feet and inches are integer variables
➤ %d is a placeholder (format specifier) for an int variable
➤ Program output:

6 feet = 72 inches
The `printf()` function is used to display output on the screen.

The `scanf()` function is used to read input from the keyboard.

`scanf()` is also defined in `stdio.h`.
USING `scanf()`

- `scanf()` reads in data and stores it in one or more variables

```c
int userAge;
scanf(" %d", &userAge);
```
**scanf() USAGE**

- The first argument (the *control string*) contains a series of placeholders.
  - These are like the ones that `printf()` uses:
    - `%d = int`, `%f = float`, `%c = char`, etc.

- Spaces are used to separate placeholders and absorb whitespace.
  - "`%d`" absorbs leading spaces and reads an integer value.
scanf() USAGE, PT. 2

- The remaining arguments to scanf() are a comma-separated list of variable names
  - Input is stored in these variables
  - Each variable name is preceded by &
  - &i means “the memory address associated with variable i”
  - We’ll talk more about this later on
```c
int a, b, c;
char d;

scanf(" %d %d", &a, &b);
scanf(" %c %d", &d, &c);
```