COMP1005
Programming and Algorithms

Functions Revisited

Jamie Twycross

Overview

- Function parameters
- Function return values
- `main`
- `printf`

Function Definition

- **Function definition syntax**:

```
return-type function-name(parameter declarations)
{
    variable declarations

    statements
}
```

Function Parameters

```
return-type function-name(parameter declarations)
{
    variable declarations

    statements
}
```

- Function **parameters** allow us to **pass values** from one function to another
- Function parameters are **automatic variables**:
  - initialised with a **copy** of value passed
- This is called **pass by value**
- What does this mean in terms of **scope** and **lifetime**?

Function Arguments

- **Function call syntax**:

```
return-type function-name(parameter declarations);
```

- A function **cannot change** the values of it arguments
- Changing the value of a parameter inside a function will **not** change the value of the argument (variable) in the calling function

Function Return Values

```
return-type function-name(parameter declarations)
{
    variable declarations

    statements
}
```

- Functions can return a **single value**
- Caller can use the returned value just like any other value
- Declare the **type** of the return value in the function definition
- Use `void` keyword to declare no return value
- Can also use `void` to declare no parameters

Function Return Values

- Use the `return` statement to return a value:

  ```
  return <value>;
  ```

- Stops the function at that point and returns to the caller, passing the value back
- A return statement can be **anywhere** in the function, not just at the end
- **Good practice** to have a return at end of function block
- Can use `return;` to exit a `void` function
- In ANSI C, default return value if not specified is `int`
- But **good practice** to always specify (removed from C99)
- Calling function does not need to do anything (or even assign) returned value

`main()` Function

```
int main(int argc, char **argv)
{
    /* code goes here */

    return 0;
}
```

- `main()` is the **program entry point**
- Also often the **program exit point**
- When reach end of `main()`, program exits and returns to OS
- `main()` returns the **exit code** of the program to the OS:
  - 0 (zero) = exited with no errors
  - non-zero (often 1) = exited with errors
- **Good practice** to put `return 0;` at end of `main()`

`printf()` Function

```
printf(char *format_string, arg1, arg2, arg3, ...);
```

- Use `printf()` to output to the screen
- Or more strictly, **standard output `stdout`**
- Part of the C **standard input/output library** `stdio`:

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

- The **f** stands for print **f**ormatted
- `printf()` takes one or more arguments:
  - A string containing **text** to print and **conversion specifiers**: `format_string`
  - A series of values to insert into the string (`arg1`, `arg2`, ...)

`printf()` Format String

- **Regular characters** in the format string are copied directly to the terminal
- Format string can also contain **conversion specifications**:
  - Begin with a `%` character
  - End with a **conversion character**
  - Other characters possible between `%` and conversion character
- A conversion specification converts and prints the **next successive argument** to `printf()`
- Possible to be more specific about how things are formatted: leading zeros, precision, ...
- Expressed in the conversion specification, e.g. `%5d` - always print 5 characters

printf() Example

```
printf("Colour %s, Decimal %d, Float %0.2f, Char %c\n", \
    "red", 123, 3.14, 'a');
```

Conversion Specifiers

Specifier	Output	Type
`%d`	decimal	`int`
`%i`	decimal	`int`
`%o`	octal	`int`
`%x %X`	hexadecimal	`int`
`%u`	unsigned decimal	`unsigned int`
`%c`	character	`char`
`%s`	string	`char *`
`%f`	floating point	`double`
`%e %E`	exponent form	`double`
`%g %G`	decimal	`double`
`%p`	pointer	`void *`
`%%`	%	N/A

Escape Sequences

- Format string can contain **escape sequences**
- Used to print **special characters**
- Begin with a \ character (**backslash**) followed by one or more characters:
  - `\n` - return
  - `\t` - tab
  - `\b` - backspace
  - `\"` - double quote
  - `\\` - backslash

Summary

- Function **parameters** and **arguments**:
  - **automatic**
  - **pass by value**
- Function **return values**: `return`
- `main()` function:
  - program **exit code**
- `printf()` function:
  - **conversion specification**
  - **escape characters**

Activities

- Read about functions in **K&R Chapter 4.1-4.4**
- Read about `printf()` in **K&R Chapter 7.2**
- Read about **escape sequences** in **K&R Chapter 2.3**
- Look at man page for `printf()`:

  ```
  $ man 3 printf
  ```

- Look at `printf()` and escape sequence pages on Wikipedia:

  http://en.wikipedia.org/wiki/Printf_format_string

  https://en.wikipedia.org/wiki/Escape_sequences_in_C