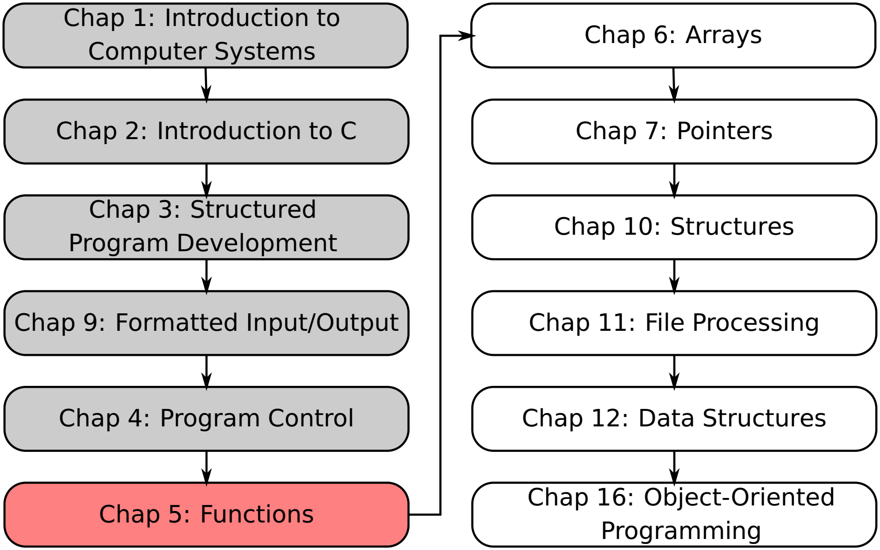
Computer Programming 143 – Lecture 11
Functions II

Electrical and Electronic Engineering Department
University of Stellenbosch

Prof Johan du Preez
Mr Callen Fisher
Dr Willem Jordaan
Dr Hannes Pretorius
Mr Willem Smit

Copyright & Disclaimer

Module Overview


Chap 1: Introduction to Computer Systems
Chap 2: Introduction to C
Chap 3: Structured Program Development
Chap 9: Formatted Input/Output
Chap 4: Program Control
Chap 5: Functions
Chap 6: Arrays
Chap 7: Pointers
Chap 10: Structures
Chap 11: File Processing
Chap 12: Data Structures
Chap 16: Object-Oriented Programming

Lecture Overview

5.1-5.6 Review of Functions I

What are functions?

A function is a piece of code or a module that

- Has been “packaged” as a unit
- Usually serve a single function

The components of a function

- A body of code to be executed
- Arguments that are passed to the function (input/data)
- A value that is returned (output/result)

5.1-5.6 Review of Functions II

How does a program execute the code in a function?

- Function calls

```
x = cos( 1.15 );
```

  - Provide function name and arguments
  - Function performs operations or manipulations
  - Function returns a result

5.1-5.6 Review of Functions III

Function definition format

return_value_type function_name(argument_list) {
  declarations;
  statements;
  return control;
}

- Example:

```
int maximum( int x, int y, int z ) {
   int max = x;       // assume x is largest
   if ( y > max )     // if y > max, then max = y
      max = y;
   if ( z > max )     // if z > max, then max = z
      max = z;
   return max;        // max is largest value
}
```

5.8 Header Files I

Header files

- Contain function prototypes for library functions
- **`stdlib.h`**, **`math.h`** , etc.
- Load with **`#include`** <*filename*>

```
#include <math.h>
```

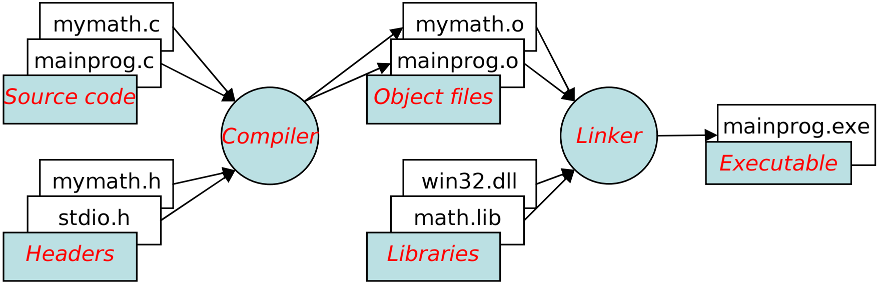
User-defined header files

- Create file with function definitions (source code) and save as **`filename.c`**
- Create file with function prototypes (header) and save as **`filename.h`**
- Use functions in a program by including

```
#include "filename.h"
```

- Reuse functions

5.8 Header Files II



5.8 Header Files III

Standard library header	**Explanation**
`<assert.h>`	Contains macros and information for adding diagnostics that aid program debugging.
`<ctype.h>`	Contains function prototypes for functions that test characters for certain properties, and function prototypes for functions that can be used to convert lowercase letters to uppercase letters and vice versa.
`<errno.h>`	Defines macros that are useful for reporting error conditions.
`<float.h>`	Contains the floating point size limits of the system.
`<limits.h>`	Contains the integral size limits of the system.
`<locale.h>`	Contains function prototypes and other information that enables a program to be modified for the current locale on which it is running. The notion of locale enables the computer system to handle different conventions for expressing data like dates, times, dollar amounts and large numbers throughout the world.
`<math.h>`	Contains function prototypes for math library functions.

5.8 Header Files IV

Standard library header	Explanation
`<setjmp.h>`	Contains function prototypes for functions that allow bypassing of the usual function call and return sequence.
`<signal.h>`	Contains function prototypes and macros to handle various conditions that may arise during program execution.
`<stdarg.h>`	Defines macros for dealing with a list of arguments to a function whose number and types are unknown.
`<stddef.h>`	Contains common definitions of types used by C for performing certain calculations.
`<stdio.h>`	Contains function prototypes for the standard input/output library functions, and information used by them.
`<stdlib.h>`	Contains function prototypes for conversions of numbers to text and text to numbers, memory allocation, random numbers, and other utility functions.
`<string.h>`	Contains function prototypes for string processing functions.
`<time.h>`	Contains function prototypes and types for manipulating the time and date.

5.9 Calling Functions: Call-by-Value and Call-by-Reference I

Call-by-value

- Copy of argument passed to function
  e.g.: `printf("Hello %d",x);`
- Changes to the variable in function do not affect original variable
- Use when function does not need to modify argument
- Avoids accidental changes to the variable

Call-by-reference

- Passes memory address of original argument
  e.g.: `scanf("%d",&x);`
- Changes to the variable in function affect original argument
- Only used with trusted functions

5.9 Calling Functions: Call-by-Value and Call-by-Reference II

For now, we focus on call-by-value

5.10 Random Number Generation I

Random number generation in C

- Use C Standard Library function **rand()** from `<stdlib.h>` header

```
i = rand();
```

  - Generates a pseudorandom number between 0 and `RAND_MAX`
  - `RAND_MAX` - a symbolic constant defined in `stdlib.h` and is equal to 2147483647
- Example of use:

```
i = 1 + rand() % 6;
```

  - Generates a pseudorandom number from 1 to 6
- Refer to the example in Fig. 5.11 in Deitel & Deitel

5.10 Random Number Generation II

Random number generation in C (cont'd...)

- Function **rand()** generates the same sequence of numbers for the same seed
- To randomise the sequence, change the seed by using function **srand()**

```
unsigned seed;
scanf( "%u", &seed );
srand( seed );
```

- To randomise without entering a seed every time

```
srand( time( NULL ) );
```

  - Reads the computer clock and passes it as seed to **srand()**
  - Requires the <time.h> header
- Refer to the example in Fig. 5.13 in Deitel & Deitel

Perspective

Today

Functions II

- Review of functions
- Header files
- Calling functions: call-by-value and call-by-reference
- Random number generation

Next lecture

Functions III

- Example: a game of chance
- Storage classes
- Scope rules

Homework

1. Study Sections 5.8-5.10 in Deitel & Deitel
2. Do Self Review Exercises 5.1(i)&(j), 5.7 in Deitel & Deitel
3. Do Exercises 5.11, 5.13, 5.14 in Deitel & Deitel