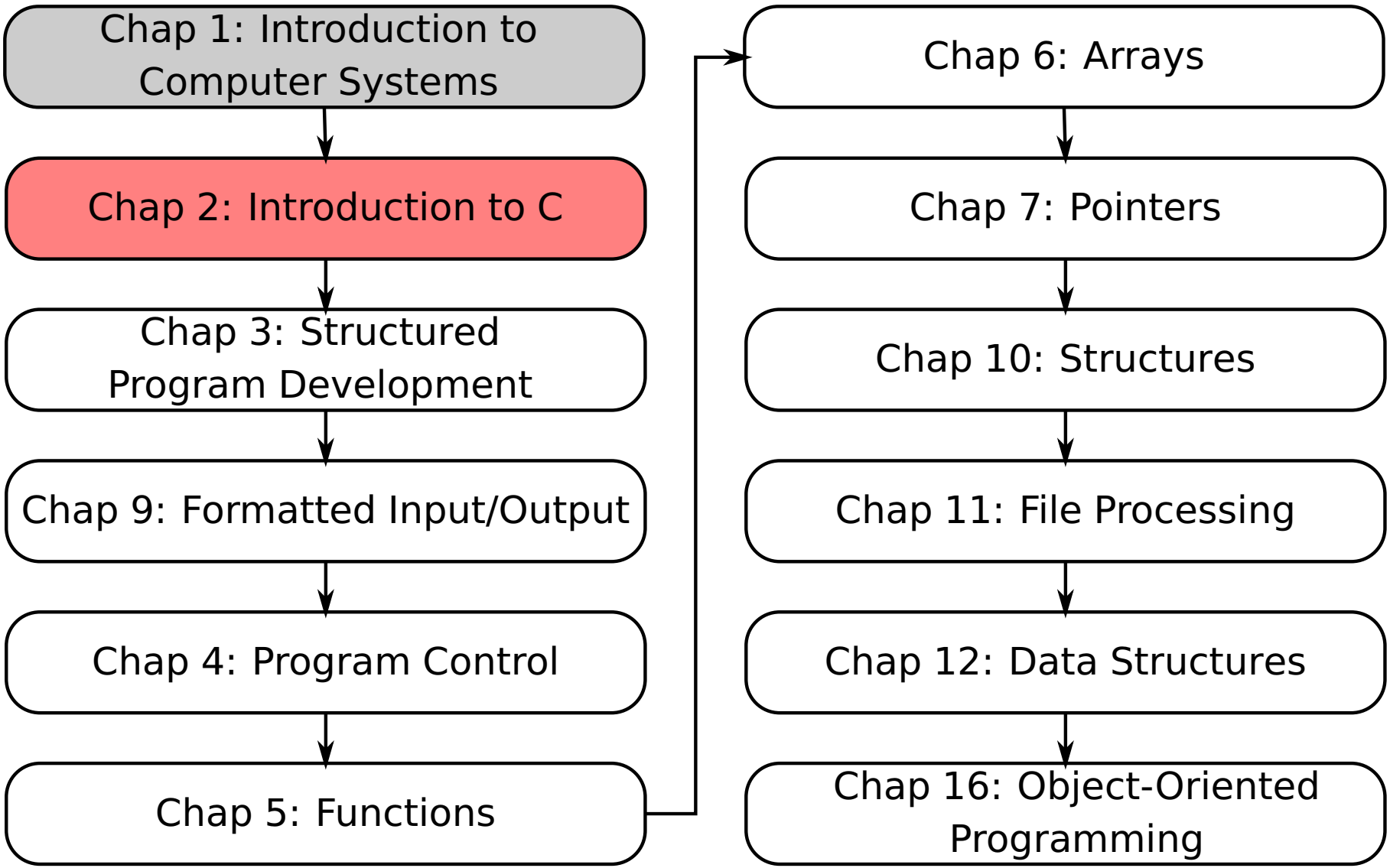
Computer Programming 143 – Lecture 3
Introduction to Programming in C

Electrical and Electronic Engineering Department
University of Stellenbosch

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

Module Overview



Lecture Overview

1. Simple C Program 1: Printing Text (2.2)
2. Simple C Program 2: Adding Two Integers (2.3)
3. Memory Concepts (2.4)
4. Arithmetic (2.5)
5. Simple C Program 3: User Input (2.3)

2.2 Simple C Program: Printing Text I

Example with escape sequences

```c
#include <stdio.h>

int main()
{
   printf( "Welcome\nto\nC!\n" );
   return 0;
}
```

Output

```
Welcome
to
C!
```

2.2 Simple C Program: Printing Text II

Some common escape sequences (Also see Chapter 9)

Escape Sequence	Description
\n	Newline. Position the cursor at the beginning of the next line.
\t	Horizontal tab. Move the cursor to the next tab stop.
\a	Alert. Sound the system bell.
\\	Backslash. Insert a backslash character in a string.
\"	Double quote. Insert a double quote character in a string.
\'	Single quote. Insert a single quote into string.
\r	Position the cursor at the beginning of the current line.
\?	Insert a question mark character.

2.3 Adding Two Integers I

```c
/* Addition program */
#include <stdio.h>
int main()
{
  int integer1;           // first number to be used
  int integer2;           // second number to be used
  int sum;                // variable in which sum will be stored

  integer1 = 45;          // assign value to integer1
  integer2 = 72;          // assign value to integer2

  printf( "First integer : %d\n", integer1 );  // print integer1 value
  printf( "Second integer : %d\n", integer2 ); // print integer2 value
  sum = integer1 + integer2;                   // assign sum
  printf( "Sum is %d\n", sum );                // print sum

  return 0;
}
```

2.3 Adding Two Integers II

Output

```
First integer : 45
Second integer : 72
Sum is 117
```

2.3 Adding Two Integers III

`int integer1, integer2, sum;`

- Declaration of variables
  - Variables: locations in memory where a value can be stored
- **`int`** means variables can hold integers (-1, 3, 0, 47)
- Variable names (identifiers)
  - `integer1, integer2, sum`
    - Identifiers: consist of letters, digits (cannot begin with a digit) and underscores(_)
    - Case sensitive
    - Should not use C keywords
- Declarations must appear before executable statements
  - Variables must be declared at the beginning/top of a code block
  - If an executable statement references and undeclared variable it will produce a syntax (compiler) error

2.3 Adding Two Integers IV

C's reserved keywords:

auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

2.3 Adding Two Integers V

printf("Sum is %d\n", sum)

- Special characters are being used
  - %d means decimal integer will be provided for printing
  - **sum** specifies what integer should be printed
- Calculations can be performed inside `printf` statements

```
printf( "Sum is %d\n", integer1 + integer2 );
```

= (assignment operator)

- Assigns a value to a variable
- Is a binary operator (has two operands)

```
sum = variable1 + variable2;
sum gets (variable1 + variable2)
```

- Variable receiving value on left

2.4 Memory Concepts I

Variables!..?

- Variables are used as temporary storage within the computer memory
- Every variable has a name, a type, a size and a value
- Variable names correspond to locations in the computer's memory
- Whenever a new value is placed into a variable, it replaces (and destroys) the previous value
- Reading variables from memory does not change them
- All variables must be declared before they can be used

2.4 Memory Concepts II

Variable declaration

```
int integer1;
int integer2, sum;
```

Variable assignments

```
integer1 = 45;
integer2 = 72;
sum = integer1 + integer2;
```

2.4 Memory Concepts III

A visual representation

Name	Value	Address
`integer1`	45	3000
`integer2`	72	3004
`sum`	117	3008

Variables

Characters	`char`	8 bits	-128 to 127
Strings	`char[]`		
Integers	**`int`**	32 bits	-2147483648 to 2147483647
Decimals	**`float`**	32 bits	about $\pm$ 1.2E-38 to $\pm$ 3.4E+38
	`double`	64 bits	about $\pm$ 2.3E-308 to $\pm$ 1.7E+308

2.5 Arithmetic I

Arithmetic calculations

- Use $*$ for multiplication and / for division
- Integer division truncates remainder
  - 7/5 evaluates to 1
- Modulus operator (%) returns the remainder
  - 7%5 evaluates to 2

Operator precedence

- Some arithmetic operators act before others (i.e., multiplication before addition)
  - Use parenthesis when needed
- Example: Find the average of three variables a, b and c
  - Do not use: `a + b + c / 3`
  - Use: `(a + b + c ) / 3`

2.5 Arithmetic II

Arithmetic operators

C operation	Arithmetic operator	Algebraic expression	C expression
Addition	+	$f + 7$	f + 7
Subtraction	-	$p - c$	p - c
Multiplication	*	$b \cdot m$	b * m
Division	/	x/y	x / y
Modulus	%	$r \mod s$	r % s

2.5 Arithmetic III

Rules of operator precedence

Operators	Operation	Order of evaluation
( )	Parentheses	Evaluated first. If the parentheses are nested, the expression in the innermost pair is evaluated first. If there are several pairs of parentheses "on the same level" (i.e., not nested), they are evaluated left to right.
`*`, `/`, `or` `%`	Multiplication, Division, Modulus	Evaluated second. If there are several, they are evaluated left to right.
`+` `or` `-`	Addition, Subtraction	Evaluated last. If there are several, they are evaluated left to right.

User input with scanf I

```c
/* Addition program */
#include <stdio.h>
int main()
{
  int integer1;    // first number to be input by user
  int integer2;    // second number to be input by user
  int sum;         // variable in which sum will be stored

  printf( "Enter first integer\n" );  // prompt
  scanf( "%d", &integer1 );           // read integer
  printf( "Enter second integer\n" ); // prompt
  scanf( "%d", &integer2 );           // read integer
  sum = integer1 + integer2;          // assign sum
  printf( "Sum is %d\n", sum );       // print sum
  return 0;
}
```

User input with scanf II

Output

```
Enter first integer
45
Enter second integer
72
Sum is 117
```

User input with scanf III

scanf("%d", &integer1);

- Obtains a value from the user
  - `scanf` uses standard input (usually keyboard)
- This scanf statement has two arguments
  - `%d` - indicates data should be a decimal integer
  - `&integer1` - location in memory to store variable
  - `&` is confusing in beginning — for now, just remember to include it with the variable name in `scanf` statements
- When executing the program the user responds to the `scanf` statement by typing in a number, and then pressing the enter (return) key

Perspective

Today

Introduction to Programming in C

- Program 1: Printing a line of text
- Program 2: Adding two numbers
- Memory concepts
- Arithmetic
- Program 3: User input

Next lecture

Structured Program Development I

- Programming decisions
- Algorithms, pseudocode and flow diagrams

Homework

1. Study Sections 2.1-2.5 in Deitel&Deitel
2. Do Self Review Exercises 2.1, 2.2, 2.4, 2.5 in Deitel&Deitel
3. Do Exercises 2.8, 2.9, 2.10, 2.13 in Deitel&Deitel