

"Everybody ... should learn to program a computer ... because it teaches you to think."

- Steve Jobs

CSE102 Computer Programming with C

2017-2018 Spring Semester

Introduction to CSE102

© 2015-2018 Yakup Genç

Largely adapted from J.R. Hanly, E.B. Koffman, F.E. Sevilgen, and others...

Portability Tips

- Because C is a hardware-independent, widely available language, applications written in C can run with little or no modifications on a wide range of different computer systems

February 2018

CSE102 Lecture 01

3

History of C

- C Programming Language
 - Evolved by Ritchie from two previous programming languages, BCPL and B
 - Used to develop UNIX
 - Used to write modern operating systems
 - Hardware independent (portable)
 - By late 1970's C had evolved to "Traditional C"
- Standardization
 - Many slight variations of C existed, and were incompatible
 - Committee formed to create a "unambiguous, machine-independent" definition
 - Standard created in 1989, updated in 1999

February 2018

CSE102 Lecture 01

2

C Standard Library

- C programs consist of pieces/modules called functions
 - A programmer can create his own functions
 - Advantage: the programmer knows exactly how it works
 - Disadvantage: time consuming
 - Programmers will often use the C library functions
 - C defines a small number of operations, instead it contains useful libraries
 - Use these as building blocks
 - Avoid re-inventing the wheel
 - If a pre-made function exists, generally best to use it rather than write your own
 - Library functions carefully written, efficient, and portable

February 2018

CSE102 Lecture 01

4

Performance Tips

- Using Standard C library functions instead of writing your own comparable versions can improve **program performance**, because these functions are carefully written to perform efficiently.

February 2018

CSE102 Lecture 01

5

Software Engineering Observation

- Read the manuals for the version of C you are using. Reference these manuals frequently to be sure you are aware of the rich collection of C features and that you are using these features correctly.

February 2018

CSE102 Lecture 01

7

Portability Tips

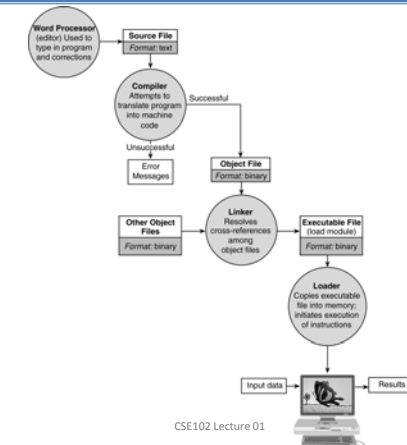
- Using Standard C library functions instead of writing your own comparable versions can improve **program portability**, because these functions are used in virtually all Standard C implementations.

February 2018

CSE102 Lecture 01

6

Typical C Program Development



February 2018

CSE102 Lecture 01

8

Software Engineering Observation

- Your computer and compiler are good teachers. If you are not sure how a feature of C works, write a sample program with that feature, compile and run the program and see what happens.

February 2018

CSE102 Lecture 01

9

Problem Requirements

- Statements of the problem
 - Understand the problem
 - Retrieve the requirements
 - Eliminate unimportant aspects
- May need to get information from specialists
- E.g. Write a program for mile to kilometer conversion

February 2018

CSE102 Lecture 01

11

Software Development

- Programming = problem solving
- Methodology
 - Specify the problem requirements
 - Analyze the problem
 - Design an algorithm
 - Implement the algorithm
 - Test and verify the program
 - Maintain and update the program

February 2018

CSE102 Lecture 01

10

Software Development

- Programming = problem solving
- Methodology
 - Specify the problem requirements
 - Analyze the problem
 - Design an algorithm
 - Implement the algorithm
 - Test and verify the program
 - Maintain and update the program

February 2018

CSE102 Lecture 01

12

Analysis

- Identify
 - Input data
 - Output data
 - Additional requirements and constraints
- Decide aspects of data
 - Representation
 - Relationships
- E.g.
 - Input: distance on miles
 - Output: distance on kilometers
 - Representation: floating point numbers
 - Relationship: 1 mile = 1.609 kilometers

February 2018

CSE102 Lecture 01

13

Designing Algorithm

- Top-down stepwise refinement
 - List major steps (sub-problems)
 - Break down each step into a more detailed list
- Desk-check your algorithm
 - Perform steps of the algorithm by yourself
- E.g.
 1. Get the distance in miles
 2. Convert the distance to kilometers
 3. Display the distance in kilometers
 - Step 2 may require further refinement
 - 2.1 The distance in kilometers is 1.609 times the distance in miles

February 2018

CSE102 Lecture 01

15

Software Development

- Programming = problem solving
- Methodology
 - Specify the problem requirements
 - Analyze the problem
 - Design an algorithm
 - Implement the algorithm
 - Test and verify the program
 - Maintain and update the program

February 2018

CSE102 Lecture 01

14

Software Development

- Methodology
 - Specify the problem requirements
 - Analyze the problem
 - Design an algorithm
 - Implement the algorithm
 - Writing the algorithm in C by converting each step into statements of C
 - Test and verify the program
 - Run the program for several input cases
 - Maintain and update the program
 - Keep the program up-to-date

February 2018

CSE102 Lecture 01

16

C Language Elements

```

/*
 * Converts distances from miles to kilometers.
 */
#include <stdio.h> /* printf, scanf definitions */
#define KMS_PER_MILE 1.609 /* conversion constant */

int main(void)
{
    double miles, /* distance in miles
    kms; /* equivalent distance in kilometers */

    /* Get the distance in miles. */
    printf("Enter the distance in miles> ");
    scanf("%lf", &miles);

    /* Convert the distance to kilometers. */
    kms = KMS_PER_MILE * miles;

    /* Display the distance in kilometers. */
    printf("That equals %f kilometers.\n", kms);

    return (0);
}

```

Diagram labels for slide 17:

- preprocessor directive: `#include <stdio.h>`
- constant: `1.609`
- reserved word: `int`, `main(void)`
- variable: `miles`, `kms`
- standard identifier: `printf`, `scanf`
- comment: `/* ... */`
- special symbol: `{`, `}`
- punctuation: `;`

February 2018

CSE102 Lecture 01

17

C Language Elements

```

/*
 * Converts distances from miles to kilometers.
 */
#include <stdio.h> /* printf, scanf definitions */
#define KMS_PER_MILE 1.609 /* conversion constant */

int main(void)
{
    double miles, /* distance in miles
    kms; /* equivalent distance in kilometers */

    /* Get the distance in miles. */
    printf("Enter the distance in miles> ");
    scanf("%lf", &miles);

    /* Convert the distance to kilometers. */
    kms = KMS_PER_MILE * miles;

    /* Display the distance in kilometers. */
    printf("That equals %f kilometers.\n", kms);

    return (0);
}

```

Diagram labels for slide 19:

- preprocessor directive: `#include <stdio.h>`
- constant: `1.609`
- reserved word: `int`, `main(void)`
- variable: `miles`, `kms`
- standard identifier: `printf`, `scanf`
- comment: `/* ... */`
- special symbol: `{`, `}`
- punctuation: `;`

February 2018

CSE102 Lecture 01

19

Preprocessor Directives

- Preprocessor modifies the text of a C program before compilation
- Preprocessor directives
 - Start with a #
- `#include <stdio.h>`
 - Each library has a header file. Include it to access the library
 - Preprocessor inserts definitions from the header
 - `stdio.h` includes information about standard input/output
- `#define KMS_PER_MILE 1.609`
 - Defines a constant macro
 - Value of `KMS_PER_MILE` can not change
 - Preprocessor replaces each occurrence of "`KMS_PER_MILE`" in the text with "`1.609`"
 - `KMS_PER_MILE` is easier to remember

February 2018

CSE102 Lecture 01

18

Function main

- C programs have exactly one main function
 - Marks the beginning of program execution
 - `(void)` indicates that function receives no data
 - `int` means that main "returns" an integer value
- Function bodies enclosed in braces `{` and `}`
 - Function body has two parts
 - Declaration
 - Executable statements

February 2018

CSE102 Lecture 01

20

Identifiers

- Reserved words
 - Ex: “int” and “void”
 - Can not be used for any other purpose
- Standard identifiers
 - Ex: scanf, printf
 - Has a special meaning but can be redefined
- User-defined identifiers
 - Ex: name of memory cells (miles) and KMS_PER_MILE
 - Free to select the name
 - Syntax rules:
 - Includes only letters, digits and underscore
 - Can not begin with digit
- C is case sensitive!

February 2018

CSE102 Lecture 01

21

Program Style

- Pick meaningful identifiers
 - Long enough to convey the meaning
- If the identifier consists of two words, place an underscore character between words
- Do not choose similar identifier names
- Use uppercase letters for names of macros
 - Use lowercase letters otherwise

February 2018

CSE102 Lecture 01

23

Reserved words

Keywords

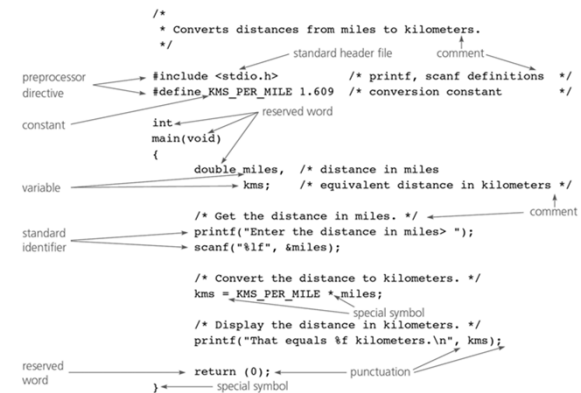
auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
const	float	short	unsigned
continue	for	signed	void
default	goto	sizeof	volatile
do	if	static	while

February 2018

CSE102 Lecture 01

22

C Language Elements



February 2018

CSE102 Lecture 01

24

Variables

- Variables: memory cells for storing data
 - Values can change
- Every variable has:
 - Name: identifier
 - Type: int, double, char
 - Size
 - Value
- Data types: abstraction of real types
 - Predefined data types
 - User-defined data types
 - Each constant or variable has a type
 - int: whole numbers (-123, 15, 27384)
 - There is a range because of finite memory cell size
 - double: real numbers (12.345, 0.5217e-4)
 - Too large and too small numbers can not be represented
 - char: character values ('a', '5', '^', ',')

February 2018

CSE102 Lecture 01

25

C Language Elements

```

/* Converts distances from miles to kilometers.
 */
#include <stdio.h> /* printf, scanf definitions */
#define KMS_PER_MILE 1.609 /* conversion constant */

int main(void)
{
    double miles, /* distance in miles */
           kms; /* equivalent distance in kilometers */

    /* Get the distance in miles. */
    printf("Enter the distance in miles> ");
    scanf("%lf", &miles);

    /* Convert the distance to kilometers. */
    kms = KMS_PER_MILE * miles;

    /* Display the distance in kilometers. */
    printf("That equals %f kilometers.\n", kms);

    return (0);
}

```

Diagram labels for the code above:

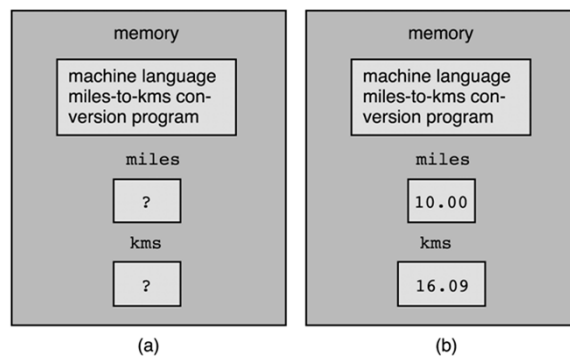
- preprocessor directive: `#include <stdio.h>`
- constant: `1.609`
- variable: `miles`, `kms`
- standard identifier: `printf`, `scanf`
- reserved word: `int`, `main`, `void`, `return`
- comment: `/* Converts distances from miles to kilometers. */`
- special symbol: `*`, `&`, `%`, `.`, `f`, `\n`
- punctuation: `{`, `}`, `;`

February 2018

CSE102 Lecture 01

27

Memory (a) Before and (b) After Execution



February 2018

CSE102 Lecture 01

26

Executable Statements

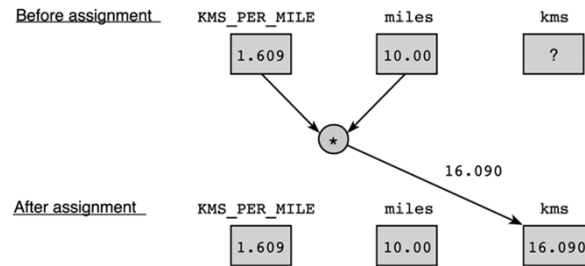
- Comes after declaration
- Compiler translates to machine language code
- Assignment statements
 - Used to store value to a variable
 - Ex: `kms = KMS_PER_MILE * miles;`
 - In general
 - `variable = expression;`
 - Assignment operator: `=`
 - Should be read as
 - becomes
 - gets
 - takes a value of
 - Previous value of variable is lost!..

February 2018

CSE102 Lecture 01

28

Effect of kms = KMS_PER_MILE * miles;



February 2018

CSE102 Lecture 01

29

Executable Statements

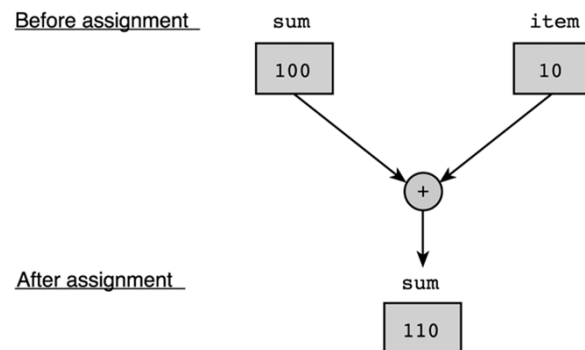
- Input/Output Operations
 - Input Operation: Reading a value into a variable by scanf
 - A different data can be entered by the user
 - Output Operation: Displaying a value by printf
- Several I/O functions in C
 - All in standard I/O library
 - #include <stdio.h>
- Function call is an executable statement.
 - Function performs the action for you.

February 2018

CSE102 Lecture 01

31

Effect of sum = sum + item;



February 2018

CSE102 Lecture 01

30

printf

Displays the output

```
printf("That equals %f kilometers.\n",kms);
```

```
printf(format string, print list);
```

- Function name: printf
- Function arguments: in parenthesis
 - Format string: "That equals %f kilometers.\n"
 - Print list: kms
- Placeholders: %c, %d, %f, %lf
- Escape sequence:
 - \n means newline : cursor moves the beginning of the next line
 - Can be used anywhere in the format string

February 2018

CSE102 Lecture 01

32

Escape sequences

Escape sequence Description

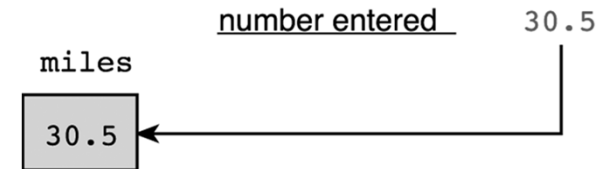
<code>\n</code>	Newline. Position the cursor at the beginning of the next line.
<code>\t</code>	Horizontal tab. Move the cursor to the next tab stop.
<code>\a</code>	Alert. Sound the system bell.
<code>\\</code>	Backslash. Insert a backslash character in a string.
<code>\"</code>	Double quote. Insert a double-quote character in a string.

February 2018

CSE102 Lecture 01

33

Effect of `scanf("%lf", &miles);`



February 2018

CSE102 Lecture 01

35

scanf

Reads the data into a variable

```
scanf("%lf", &miles);
```

```
scanf(format string, input list);
```

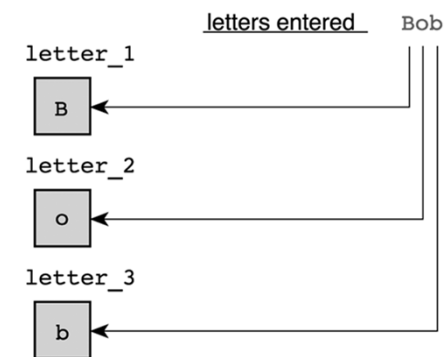
- Function name: `scanf`
- Function arguments: in paranthesis
 - Format string: `"%lf"`
 - Input list: `&miles`
- Address-of operator: `&`
 - Used to inform `scanf` about the location of the variable
 - If not used, `scanf` knows only the value of the variable

February 2018

CSE102 Lecture 01

34

Scanning Data Line Bob

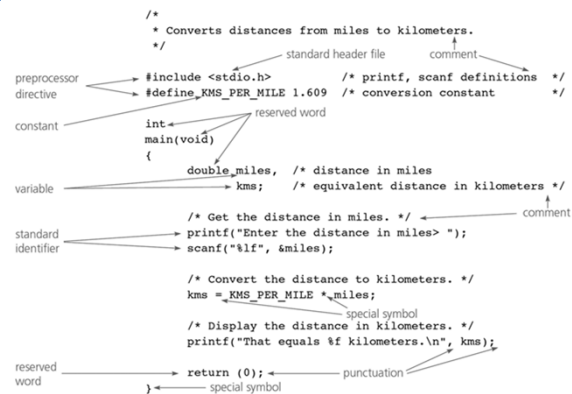


February 2018

CSE102 Lecture 01

36

C Language Elements



February 2018

CSE102 Lecture 01

37

General Form of a C Program

```

preprocessor directives
main function heading
{
    declarations
    executable statements
}

```

February 2018

CSE102 Lecture 01

39

Others

- The return statement
 - Transfers the control to the OS
 - Return value indicates whether the operation is successful or not
- Comments
 - Ignored by the compiler
 - Provides information for the programmer

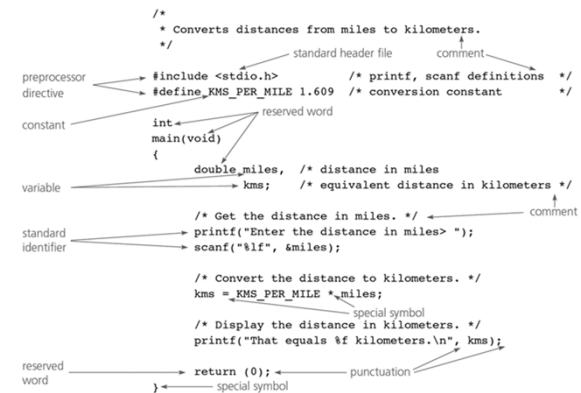
```
/* this is a comment */
```

February 2018

CSE102 Lecture 01

38

C Language Elements



February 2018

CSE102 Lecture 01

40

Program Style

- One statements in each line
- Use extra spaces for readability
 - Compiler ignores them
 - Leave a space before and after operators
 - Indent each block
 - Insert blank lines between sections
- Use comments
 - Write a descriptive comment for
 - the program (header section)
 - each identifier
 - each program section

February 2018

CSE102 Lecture 01

41

Arithmetic Expressions

- Unary Operators: +, -
 - One operand
- Assignment:
 - The value of expression is evaluated and result is assigned
 - What if the type of the expression and the type of the variable is different?
 - Assignment of int to double
 - Fractional part is zero
 - Assignment of double to int
 - Fractional part is lost
 - Automatic type conversion
 - Type casting
(int) 3.7

February 2018

CSE102 Lecture 01

43

Arithmetic Expressions

- Manipulates type int and double data
- Binary operators: +, -, *, /, %
 - Two operand: constant, variable or expression
 - Type of the result depend on the types of the operands
 - int if both operands are integer
 - double otherwise
 - Mixed type expression???
- / operator
 - Integer division: computes integral part of the division
 - Division by zero!..
- % operator
 - Returns integer remainder
 - Zero right operand? Undefined!..
 - Negative right operand is non standard

February 2018

CSE102 Lecture 01

42

Expression Evaluation

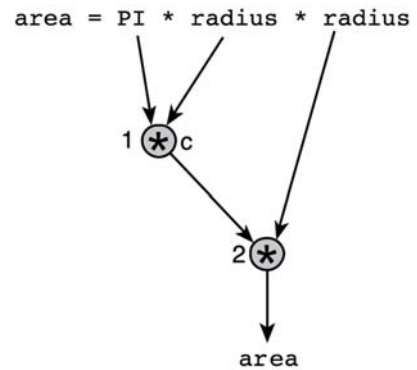
- If there are multiple operators in an expression the order of evaluation makes a difference
 - Ex: $x / y * z$
- Evaluation Rules:
 - Parenthesis rule:
 - All expressions in parenthesis are evaluated separately
 - Nested parenthesis evaluated inside out
 - Precedence rule:
 - There is an evaluation order in operators
 - Unary +, -
 - *, /, %
 - Binary +, -
 - Associativity rule:
 - Operators in the same sub-expression and at the same precedence level
 - Unary: right to left
 - Binary: left to right

February 2018

CSE102 Lecture 01

44

Evaluation Tree for $\text{area} = \text{PI} * \text{radius} * \text{radius};$

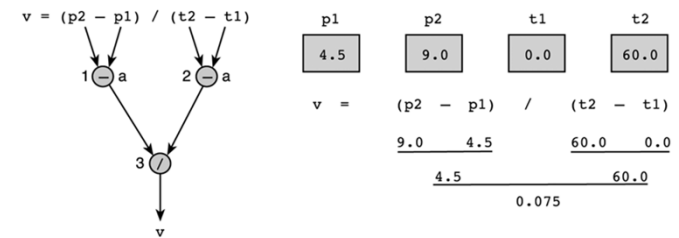


February 2018

CSE102 Lecture 01

45

Evaluation for $v = (p2 - p1) / (t2 - t1);$



February 2018

CSE102 Lecture 01

47

Step-by-Step Expression Evaluation

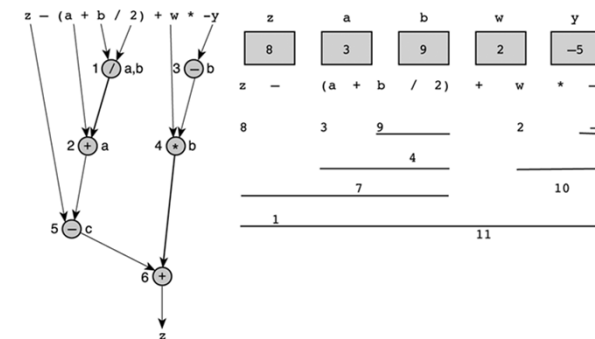
$$\begin{array}{rcl} \text{area} & = & \text{PI} * \text{radius} * \text{radius} \\ & & 3.14159 \quad 2.0 \quad 2.0 \\ & & \underline{6.28318} \\ & & 12.56636 \end{array}$$

February 2018

CSE102 Lecture 01

46

Evaluation for $z - (a + b / 2) + w * -y$



February 2018

CSE102 Lecture 01

48

Writing Mathematical Formulas

- Use parentheses as needed to specify the order of evaluation
 - Place numerator and denominator of a division in parentheses

$$m = (a - b) / (c + d)$$
 - Use extra parentheses for readability

$$(a * b * c) + (d / e) - (a + b)$$
- Do not skip `*` as in mathematical formulas
 - In math: $d = b^2 - 4ac$
 - In C: `d = b * b - 4 * a * c;`
- Two operators can be one after the other
`a * -(b + c)`

February 2018

CSE102 Lecture 01

49

Case Study: Coin Processor

- Design an algorithm
 1. Get and display the customer's initials
 2. Get the count for each kind of coin
 3. Compute the total value in cents
 4. Find the value in dollars and cents
 5. Display dollars and cents
 - Some steps need to refine!...
- Implement the algorithm
 - In the next slide
- Test and verify the program
- Maintain and update the program

February 2018

CSE102 Lecture 01

51

Case Study: Coin Processor

- Problem requirements
 - Convert change to personalized credit slip
 - User enters the number of each kind of coin
- Analyze the problem
 - Personalizing the slip: use customers initials
 - Count for each type of coin
 - Total value of the coins in dollars and cents
 - Input data
 - Initials: first, middle, last are characters
 - Counts: dollars, quarters, dimes, nickels, pennies are integers
 - Output data
 - Dollars and cents: total_dollars and change are integers
 - Intermediate data
 - Total value in cents: total_cents is integer
 - Relationships
 - total_cents =
 - total_dollars =
 - change =

February 2018

CSE102 Lecture 01

50

Supermarket Coin Value Program

```

1.  /*
2.   * Determines the value of a collection of coins.
3.   */
4.   #include <stdio.h>
5.   int
6.   main(void)
7.   {
8.       char first, middle, last; /* input - 3 initials      */
9.       int pennies, nickels; /* input - count of each coin type */
10.      int dimes, quarters; /* input - count of each coin type */
11.      int dollars; /* input - count of each coin type */
12.      int change; /* output - change amount */
13.      int total_dollars; /* output - dollar amount */
14.      int total_cents; /* total cents */
15.
16.      /* Get and display the customer's initials. */
17.      printf("Type in 3 initials and press return> ");
18.      scanf("%c%c%c", &first, &middle, &last);
19.      printf("\n%c%c%c, please enter your coin information.\n",
20.             first, middle, last);
21.
22.      /* Get the count of each kind of coin. */
23.      printf("Number of $ coins > ");
24.      scanf("%d", &dollars);
25.      printf("Number of quarters> ");

```

(continued)

February 2018

CSE102 Lecture 01

52

Supermarket Coin Value Program (cont'd)

```

26. scanf("%d", &quarters);
27. printf("Number of dimes > ");
28. scanf("%d", &dimes);
29. printf("Number of nickels > ");
30. scanf("%d", &nickels);
31. printf("Number of pennies > ");
32. scanf("%d", &pennies);
33.
34. /* Compute the total value in cents. */
35. total_cents = 100 * dollars + 25 * quarters + 10 * dimes +
36. 5 * nickels + pennies;
37.
38. /* Find the value in dollars and change. */
39. dollars = total_cents / 100;
40. change = total_cents % 100;
41.
42. /* Display the credit slip with value in dollars and change. */
43. printf("\n\nJRS Coin Credit\nDollars: %d\nChange: %d cents\n",
44. first, middle, last, dollars, change);
45.
46. return (0);
47. }

```

Type in 3 initials and press return> JRS
JRS, please enter your coin information.
Number of \$ coins > 2
Number of quarters > 14
Number of dimes > 12
Number of nickels > 25
Number of pennies > 131

JRS Coin Credit
Dollars: 9
Change: 26 cents

February 2018

CSE102 Lecture 01

53

Output Formatting

- Default formatting
- User-defined format
 - int: %4d (%nd)
 - Field width
 - Right justified
 - - sign included in the count
 - C expands the field width if necessary
 - double: %6.2f (%n.mf)
 - Field width
 - Decimal places
 - Decimal point, minus sign included in the field width
 - Values between -99.99 to 999.99 for %6.2f
 - At least one digit before decimal point
 - Values are rounded if there are more decimal places
 - -9.536 becomes -9.54
 - Use %d or %.3f not to have leading blanks

February 2018

CSE102 Lecture 01

55

Case Study: Coin Processor

- Test and verify the program
 - Try the program for several inputs
 - Make sure that program runs correctly
- Maintain and update the program
 - Later!...

February 2018

CSE102 Lecture 01

54

Input and Output Redirection

- Interactive mode
- Batch mode
 - Input Redirection: standard input is associated with a file instead of keyboard


```
myprog < inputfile
```

 - No need to display prompting message
 - Display the message about input (echo print)
 - Output Redirection: standard output is associated with a file instead of screen


```
myprog > outputfile
```

 - Can print the file to get the hardcopy

February 2018

CSE102 Lecture 01

56

Batch Version of Miles-to-Km Conversion Program

```

1. /* Converts distances from miles to kilometers. */
2.
3. #include <stdio.h> /* printf, scanf definitions */
4. #define KMS_PER_MILE 1.609 /* conversion constant */
5.
6. int
7. main(void)
8. {
9.     double miles, /* distance in miles */
10.    kms; /* equivalent distance in kilometers */
11.
12.    /* Get and echo the distance in miles. */
13.    scanf("%lf", &miles);
14.    printf("The distance in miles is %.2f.\n", miles);
15.
16.    /* Convert the distance to kilometers. */
17.    kms = KMS_PER_MILE * miles;
18.
19.    /* Display the distance in kilometers. */
20.    printf("That equals %.2f kilometers.\n", kms);
21.
22.    return (0);
23. }

```

The distance in miles is 112.00.
That equals 180.21 kilometers.

February 2018

CSE102 Lecture 01

57

Miles-to-Km Conversion Program with Named Files

```

1. /* Converts distances from miles to kilometers. */
2.
3. #include <stdio.h> /* printf, scanf, fprintf, fclose, fopen, fclose
4. definitions */
5. #define KMS_PER_MILE 1.609 /* conversion constant */
6.
7. int
8. main(void)
9. {
10.     double miles, /* distance in miles */
11.    kms; /* equivalent distance in kilometers */
12.     FILE *inp, /* pointer to input file */
13.    *outp; /* pointer to output file */
14.
15.    /* Open the input and output files. */
16.    inp = fopen("distance.dat", "r");
17.    outp = fopen("distance.out", "w");
18.
19.    /* Get and echo the distance in miles. */
20.    fscanf(inp, "%lf", &miles);
21.    fprintf(outp, "The distance in miles is %.2f.\n", miles);
22.
23.    /* Convert the distance to kilometers. */
24.    kms = KMS_PER_MILE * miles;
25.
26.    /* Display the distance in kilometers. */
27.    fprintf(outp, "That equals %.2f kilometers.\n", kms);
28.
29.    /* Close files. */
30.    fclose(inp);
31.    fclose(outp);
32.
33.    return (0);
34. }

```

Contents of input file distance.dat
112.0

Contents of output file distance.out
The distance in miles is 112.00.
That equals 180.21 kilometers.

February 2018

CSE102 Lecture 01

59

Use of Input/Output Files

- C allows to explicitly name an input or output file
 - Declaring file pointer


```
FILE * inp;
FILE * outp;
```
 - Opening file


```
inp = fopen("filename", "r");
outp = fopen("filename", "w");
```
 - Reading from a file


```
fscanf(inp, "%d", &nickels);
```
 - Writing to a file


```
fprintf(outp, "Total is %d \n", value);
```
 - Closing file


```
fclose(inp);
```

February 2018

CSE102 Lecture 01

58

Case Studies:

- Compute change for a given amount of money

February 2018

CSE102 Lecture 01

60

Programming Errors

- Error = bug
- Process of correcting errors: debugging
- Error messages
 - Depends on the system used
 - Not always easy to understand
- Three kind of errors:
 - Syntax errors
 - Violation of grammar rule
 - Detected by the compiler
 - Run-time errors
 - Detected while execution
 - Illegal operation, division by zero etc.
 - Logic errors
 - Program runs but produces incorrect result

February 2018

CSE102 Lecture 01

61

A Program with a Run-Time Error

```

111 #include <stdio.h>
112
113 int
114 main(void)
115 {
116     int    first, second;
117     double temp, ans;
118
119     printf("Enter two integers> ");
120     scanf("%d%d", &first, &second);
121     temp = second / first;
122     ans = first / temp;
123     printf("The result is %.3f\n", ans);
124
125     return (0);
126 }

```

Enter two integers> 14 3
Arithmetic fault, divide by zero at line 272 of routine main

February 2018

CSE102 Lecture 01

63

A Program with Syntax Errors

```

221 /* Converts distances from miles to kilometers. */
222
223 #include <stdio.h>          /* printf, scanf definitions */
224 #define KMS_PER_MILE 1.609 /* conversion constant */
225
226 int
227 main(void)
228 {
229     double kms;
230
231     /* Get the distance in miles. */
232     printf("Enter the distance in miles> ");
233     ***** Semicolon added at the end of the previous source line
234     scanf("%lf", &miles);
235     ***** Identifier "miles" is not declared within this scope
236     ***** Invalid operand of address-of operator
237
238     /* Convert the distance to kilometers. */
239     kms = KMS_PER_MILE * miles;
240     ***** Identifier "miles" is not declared within this scope
241
242     /* Display the distance in kilometers. */
243     printf("That equals %f kilometers.\n", kms);
244     return (0);
245 }
246 ***** Unexpected end-of-file encountered in a comment
247 ***** ")" inserted before end-of-file

```

February 2018

CSE102 Lecture 01

62

Revised Coin Value Program

```

1. int
2. main(void)
3. {
4.     char first, middle, last; /* input - 3 initials */
5.     int pennies, nickels; /* input - count of each coin type */
6.     int dimes, quarters; /* input - count of each coin type */
7.     int dollars; /* input - count of each coin type */
8.     int change; /* output - change amount */
9.     int total_dollars; /* output - dollar amount */
10.    int total_cents; /* total cents */
11.    int year; /* input - year */
12.
13.    /* Get the current year. */
14.    printf("Enter the current year and press return> ");
15.    scanf("%d", &year);
16.
17.    /* Get and display the customer's initials. */
18.    printf("Type in 3 initials and press return> ");
19.    scanf("%c%c%c", &first, &middle, &last);
20.    printf("\n%c%c%c, please enter your coin information for %d.\n",
21.        first, middle, last, year);
22.    ...

```

February 2018

CSE102 Lecture 01

64

A Program That Produces Incorrect Results

```
1. #include <stdio.h>
2.
3. int
4. main(void)
5. {
6.     int    first, second, sum;
7.
8.     printf("Enter two integers> ");
9.     scanf("%d%d", first, second); /* ERROR!! should be  &first, &second */
10.    sum = first + second;
11.    printf("%d + %d = %d\n", first, second, sum);
12.
13.    return (0);
14. }
```

Enter two integers> 14 3
5971289 + 5971297 = 11942586