Week 2 - Lecture 1

Control Statements – Selection

Edited by: Dr. Wooi Ping Cheah

Autumn 2022

Overview

- Write a C Program
- Compile and Run a C Program
- Language Insecurities
- Practice Hygienic Coding

Program Structure

- All C programs require a main function.

```
2  void main(void)
3  {
4
5  }
```

- Without it …

```
c:/mingw/bin/../lib/gcc/mingw32/5.3.0/../../../libmingw32.a(main.o):(.text.startup+0xa0):
undefined reference to `WinMain@16'
collect2.exe: error: ld returned 1 exit status
```

- Too many of it …

```
C:\Users\z2017233\Desktop>gcc lecture2.c -o lecture2
lecture2.c:8:6: error: redefinition of 'main'
 void main(void)
      ^
lecture2.c:2:6: note: previous definition of 'main' was here
 void main(void)
      ^
```

Program Structure (2)

- C programs contain one or more functions, one of which *must* be `main`.

- Every program in C begins executing at the function `main`.

- The keyword void/`int` to the left of `main` indicates that `main` "returns" nothing/an integer (whole number) value.

Program Structure (3)

Program starts at the beginning of main. A left brace, {, begins the body of code, whereas a corresponding right brace ends.

```
 8  #include <stdio.h>
 9
10  int main(void)
11  {
12      int number = 0;
13
14      printf("Current number is: %d\n", number);
15
16      printf("Please enter a new number: ");
17      scanf("%d", &number);
18
19      printf("You've entered: %d\n", number);
20
21      return 0;
22
23  }
```

- This pair of braces and the portion of the program between the braces is called a block

Return Value

- It is common to return 0 to indicate that the program has run and exited successfully.

```c
#include <stdio.h>

int main(void)
{
    int number = 0;

    printf("Current number is: %d\n", number);

    printf("Please enter a new number: ");
    scanf("%d", &number);

    printf("You've entered: %d\n", number);

    return 0;

}
```

```c
void main(void)
{

}
```

Return Value (2)

- A program can have multiple functions.
- Each function may or may not return a value.

```
26  #include <stdio.h>
27
28  void myPrint(void);
29  int myReturn(void);
30
31  int main(void)
32  {
33      int number = 0;
34
35      myPrint();
36
37      printf("Current number is: %d\n", number);
38
39      number = myReturn();
40
41      printf("The number is now: %d\n", number);
42
43      return 0;
44
45  }
46
47  void myPrint(void)
48  {
49      printf("Hello There !!\n");
50  }
51
52  int myReturn(void)
53  {
54      return 5;
55  }
```

```
C:\Users\z2017233\Desktop>lecture2
Hello There!!
Current number is: 0
The number is now: 5

C:\Users\z2017233\Desktop>_
```

Libraries

```
2   void main(void)
3   {
4
5   }
```

- Not all programs need libraries.
- Example libraries:
  - limits.h
  - math.h
  - stdio.h

```
8   #include <stdio.h>
9
10  int main(void)
11  {
12      int number = 0;
13
14      printf("Current number is: %d\n", number);
15
16      printf("Please enter a new number: ");
17      scanf("%d", &number);
18
19      printf("You've entered: %d\n", number);
20
21      return 0;
22
23  }
```

- Without it …

```
C:\Users\z2017233\Desktop>gcc lecture2.c -o lecture2
lecture2.c: In function 'main':
lecture2.c:13:2: warning: implicit declaration of function 'printf' [-Wimplicit-function-declaration]
  printf("Current number is: %d", number);
  ^
lecture2.c:13:2: warning: incompatible implicit declaration of built-in function 'printf'
lecture2.c:13:2: note: include '<stdio.h>' or provide a declaration of 'printf'
lecture2.c:16:2: warning: implicit declaration of function 'scanf' [-Wimplicit-function-declaration]
  scanf("%d%", &number);
  ^
lecture2.c:16:2: warning: incompatible implicit declaration of built-in function 'scanf'
lecture2.c:16:2: note: include '<stdio.h>' or provide a declaration of 'scanf'
```

Variables and Data Types

- Variables must be declared before first use.

```c
#include <stdio.h>

int main(void)
{
    int number = 0;

    printf("Current number is: %d\n", number);

    printf("Please enter a new number: ");
    scanf("%d", &number);

    printf("You've entered: %d\n", number);

    return 0;

}
```

- Otherwise …

```
C:\Users\z2017233\Desktop>gcc lecture2.c -o lecture2
lecture2.c: In function 'main':
lecture2.c:31:36: error: 'number' undeclared (first use in this function)
  printf("Current number is: %d\n", number);
                                    ^
lecture2.c:31:36: note: each undeclared identifier is reported only once for each function it appears in
```

Variables and Data Types (2)

- Variables can be initialised.

```
26  #include <stdio.h>
27
28  int main(void)
29  {
30      int counter;
31      int number = 0;
32
33      while(number < 1)
34      {
35          printf("Please enter a new number: ");
36          scanf("%d", &number);
37
38          counter = counter + 1;
39      }
40
41      if(counter > 1)
42      {
43          printf("User has entered %d new numbers\n", counter);
44      }
45      else
46      {
47          printf("User has entered %d new number\n", counter);
48      }
49
50      return 0;
51
52  }
```

```
C:\Users\z2017233\Desktop>gcc lecture2.c -o lecture2

C:\Users\z2017233\Desktop>lecture2
Please enter a new number: 1
User has entered 4194433 new numbers
```

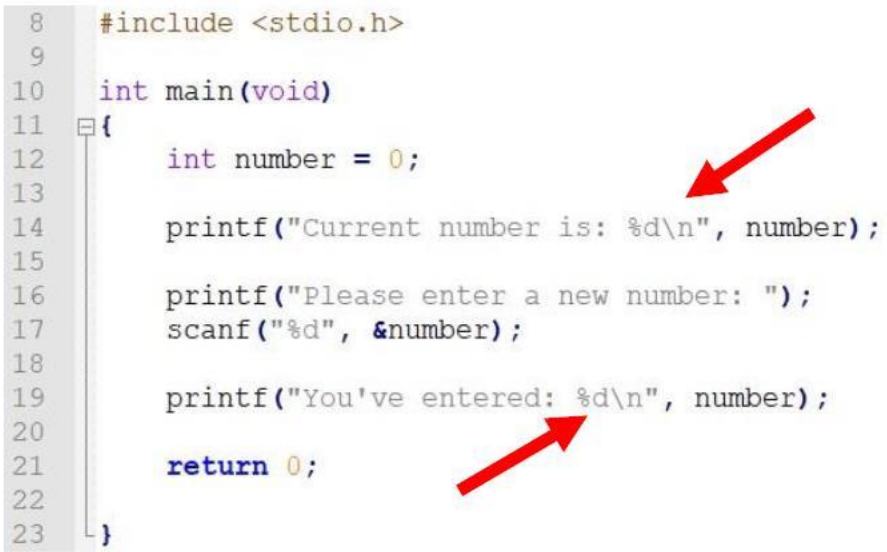
Input with scanf

- Read data from the standard input stream (stdin) and store that data in variables

```
8   #include <stdio.h>
9
10  int main(void)
11  {
12      int number = 0;
13
14      printf("Current number is: %d\n", number);
15
16      printf("Please enter a new number: ");
17      scanf("%d", &number);
18
19      printf("You've entered: %d\n", number);
20
21      return 0;
22
23  }
```

Output with printf

- Output **"formatted"** data to the standard output e.g. monitor.



```
 8  #include <stdio.h>
 9
10  int main(void)
11  {
12      int number = 0;
13
14      printf("Current number is: %d\n", number);
15
16      printf("Please enter a new number: ");
17      scanf("%d", &number);
18
19      printf("You've entered: %d\n", number);
20
21      return 0;
22
23  }
```

- Using correct format specifier is important!!

Some useful characters for Printf()

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. Produces a sound or visible alert without changing the current cursor position.
\\	Backslash. Insert a backslash character in a string.
\"	Double quote. Insert a double-quote character in a string.

Comments

- Use block or single line comment to explain what your program does.

```
149  int main(void)
150  {
151      /* This program calculate the remainder if division,
152         and return zero to the shell */
153
154      int i = (10 % 3);
155
156      // The line belows return zero to the shell that calls the program
157      return 0;
158  }
```

Overview

- Write a C Program
- **Compile and Run a C Program**
- Language Insecurities
- Practice Hygienic Coding

Compile a C Program

- gcc filename.c –o filename

Compile and link
i.e. reference to functions
e.g. standard libraries

Create an executable called “filename”

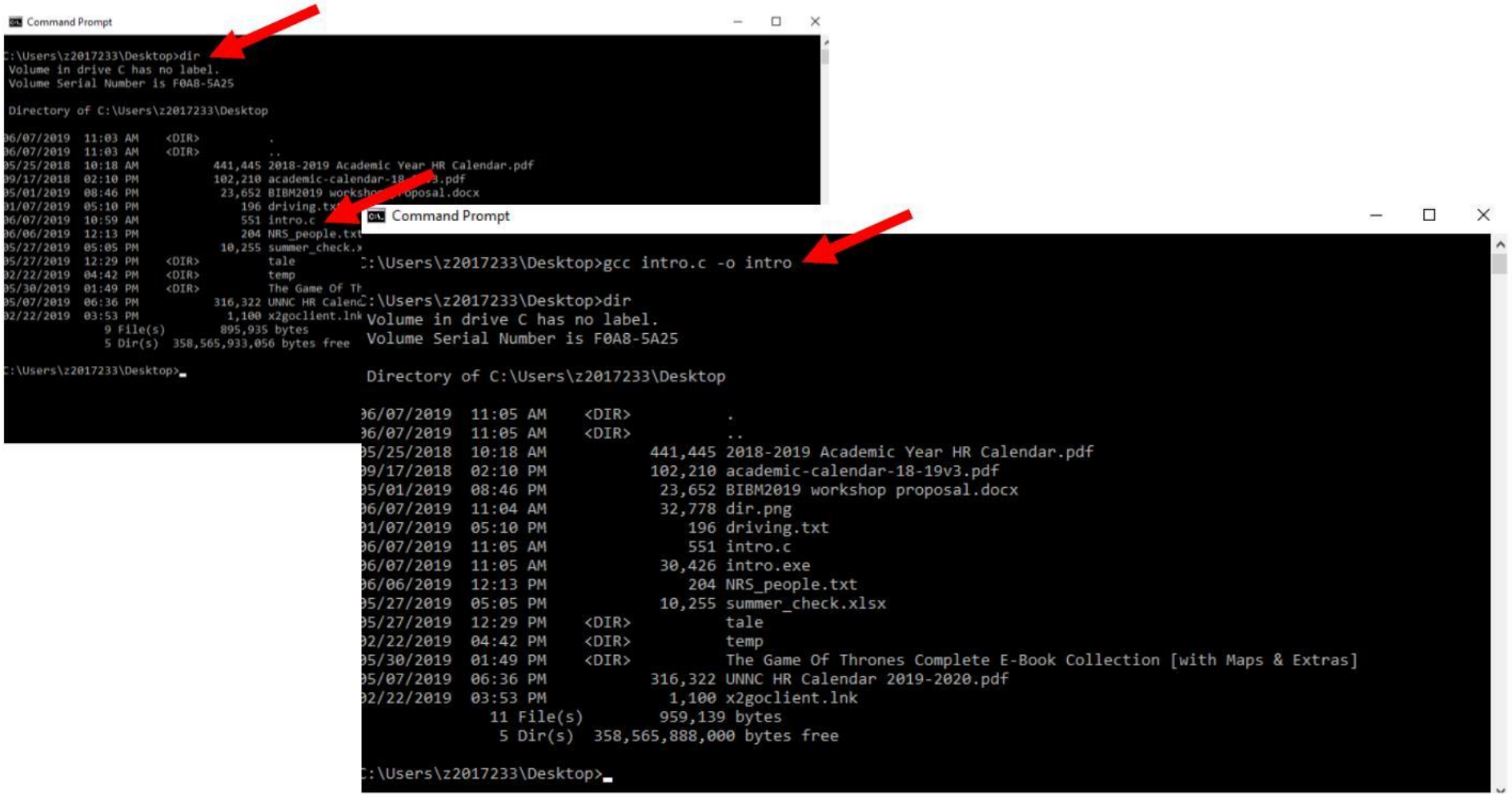
DO NOT type in filename.c !!!!!

```
C:\Users\z2017233\Desktop>gcc selection.c -o selection.c
gcc: fatal error: input file 'selection.c' is the same as output file
compilation terminated.
```

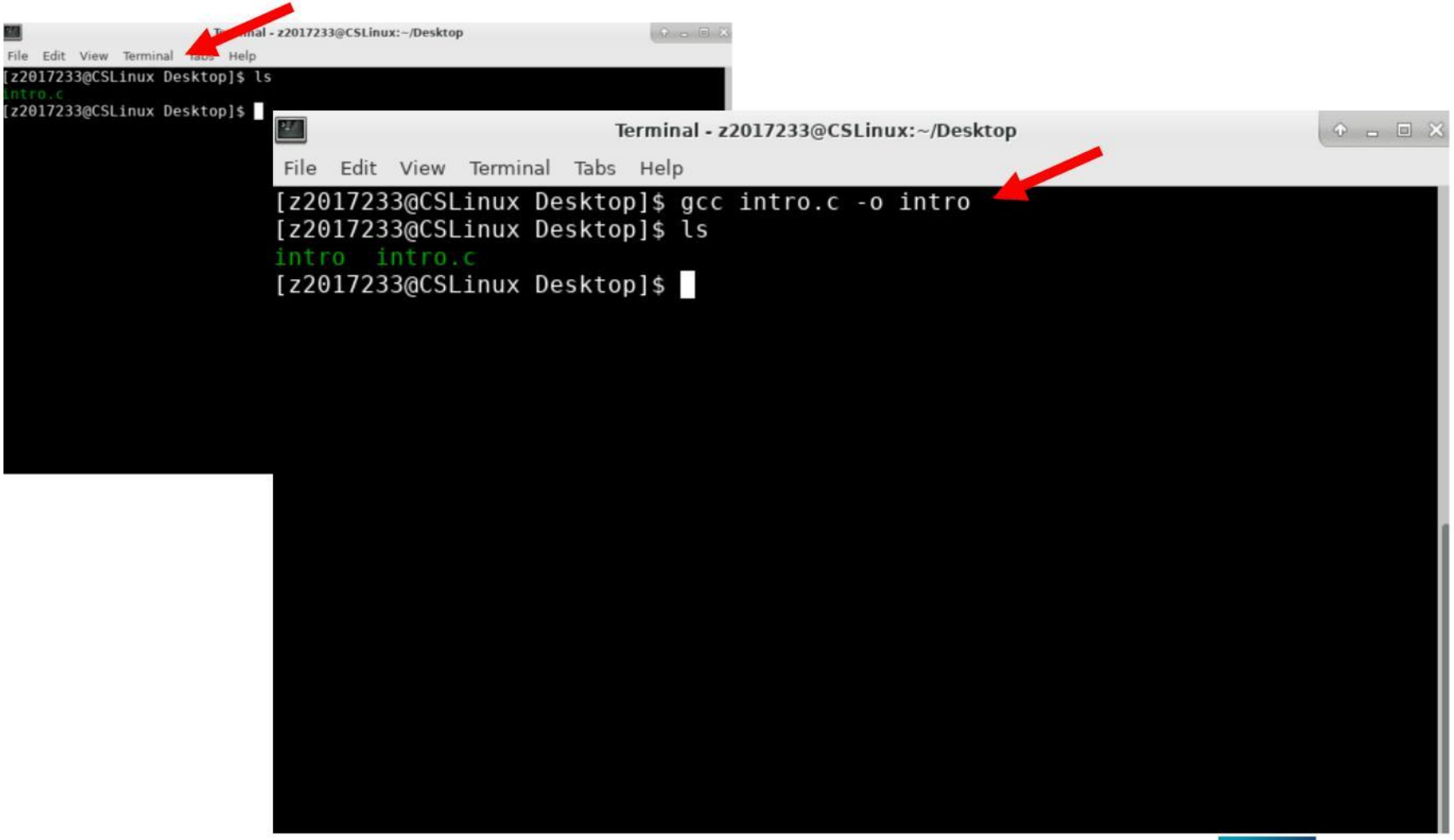
- gcc filename.c

Create “default” an executable call a.out

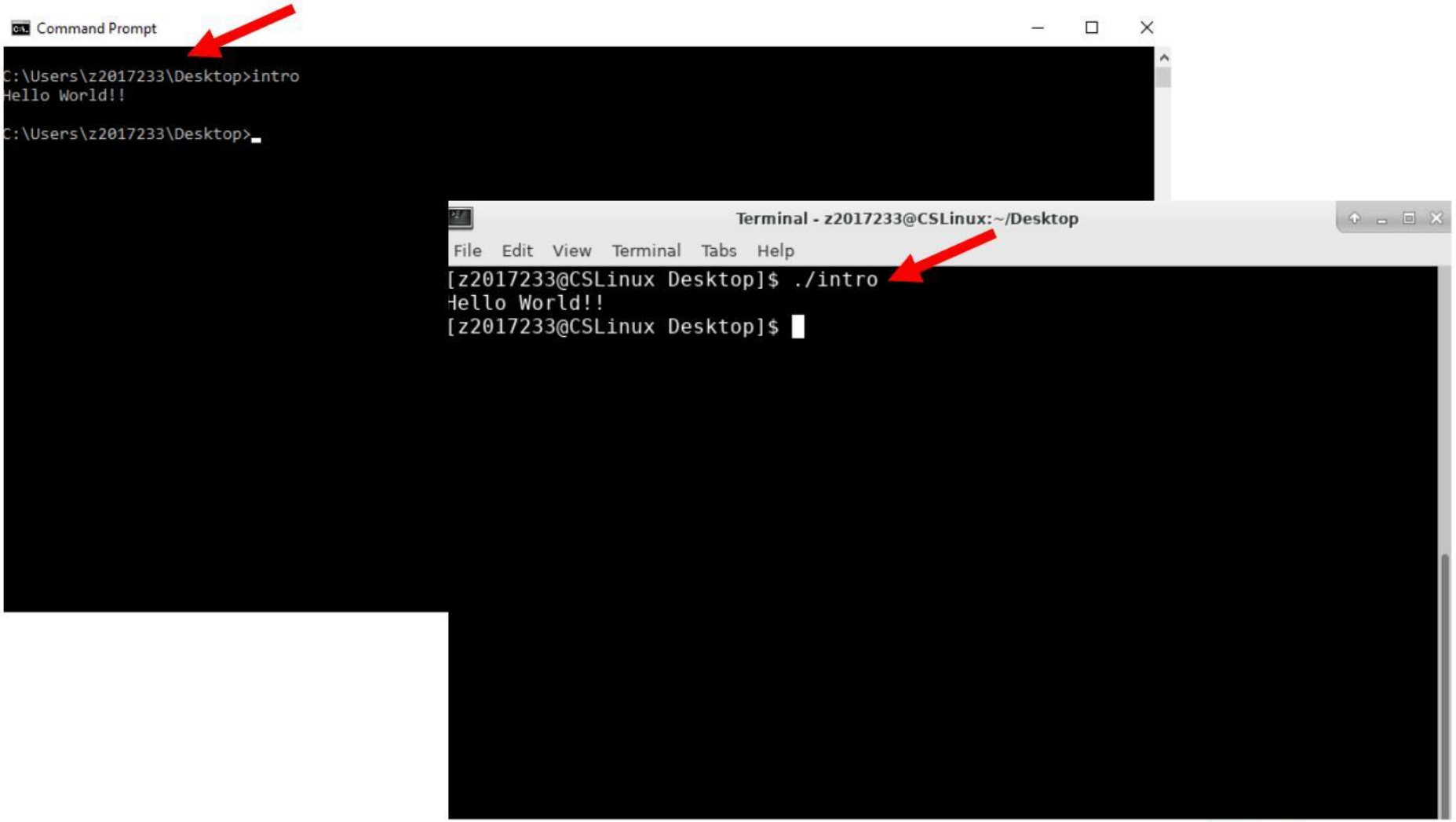
Compile a C Program - Windows



Compile a C Program - Linux



Run a C Program

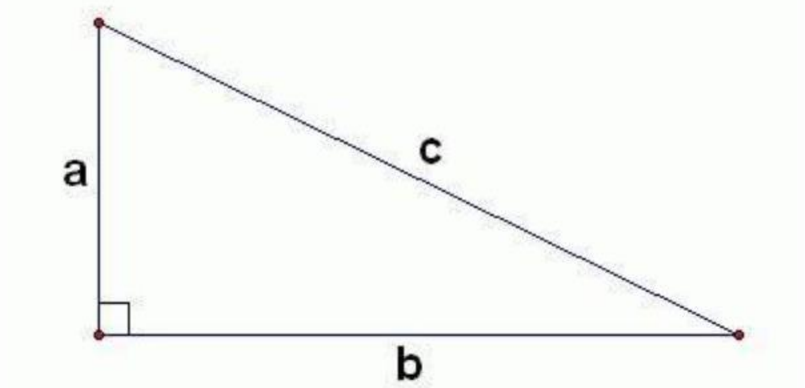


Example 1: Sum

```
1  // Fig. 2.5: fig02_05.c
2  // Addition program.
3  #include <stdio.h>
4
5  // function main begins program execution
6  int main( void )
7  {
8     int integer1; // first number to be entered by user
9     int integer2; // second number to be entered by user
10    int sum; // variable in which sum will be stored
11
12    printf( "Enter first integer\n" ); // prompt
13    scanf( "%d", &integer1 ); // read an integer
14
15    printf( "Enter second integer\n" ); // prompt
16    scanf( "%d", &integer2 ); // read an integer
17
18    sum = integer1 + integer2; // assign total to sum
19
20    printf( "Sum is %d\n", sum ); // print sum
21 } // end function main
```

Example 2: Right-Angled Triangle

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

int main(int argc, char *argv[])
{
    int x, y, z;

    printf("Enter value for x: ");
    scanf("%d", &x);
    if(x < 1)
    {
        printf("Invalid value\n");
        exit(1);
    }

    printf("Enter value for y: ");
    scanf("%d", &y);
    if(y < 1)
    {
        printf("Invalid value\n");
        exit(1);
    }

    printf("Enter value for z: ");
    scanf("%d", &z);
    if(z < 1)
    {
        printf("Invalid value\n");
        exit(1);
    }

    int lhs = x * x + y * y;
    int rhs = z * z;

    if(lhs == rhs)
    {
        printf("Right angled triangle\n");
    }
    else
    {
        printf("Not right angled, %d does not equal %d\n", lhs, rhs );
    }

}
```



$a^2 + b^2 = c^2$

Source: https://mathblog.com/reference/theorems/pythagorean-theorem/

Overview

- Write a C Program
- Compile and Run a C Program
- **Language Insecurities**
- Practice Hygienic Coding

Language Insecurities

- Style and expressiveness
- Valid typos
- Error detection
- Misunderstanding the language
- Wrong expectations
- Run-time error detection

Style and Expressiveness

- How clearly the language constructs can "express" the developer's intentions.
- For example, switch statement cases must end with break, return, or a comment indicating a fall-through

```c
#include <stdio.h>

void doSomething();
void doSomethingElse();
void doDefaultThing();

int main()
{
    int value = 0;

    switch(value)
    {
        case 1:
            doSomething();

        case 2:
            doSomethingElse();
            break;

        default:
            doDefaultThing();
            break;
    }
}
```

```c
#include <stdio.h>

void doSomething();
void doSomethingElse();
void doDefaultThing();

int main()
{
    int value = 0;

    switch(value)
    {
        case 1:
            doSomething();
            /* falls through */

        case 2:
            doSomethingElse();
            break;

        default:
            doDefaultThing();
            break;
    }
}
```

Valid Typos

```c
#include <stdio.h>
int main()
{
    int a = 0;
    if(a = 1)
        printf("a is NOT equal to zero\n");
    else
        printf("a is equal to zero\n");

    return 0;
}
```

```c
#include <stdio.h>
int main()
{
    int a = 0;
    if(a == 1)
        printf("a is NOT equal to zero\n");
    else
        printf("a is equal to zero\n");

    return 0;
}
```

Error Detection

```c
#include <stdio.h>
int main()
{
    int b = 1.25;
    double c = 1.25;

    printf("The sum of b and c is %.2f\n", (c+b));

    return 0;
}
```

Understanding the Language

```
1  #include <stdio.h>
2  int main()
3  {
4      int d = 1;
5      int e = 2;
6      int f = 3;
7      printf("%d\n", (d+e*f));
8      printf("%d\n", ((d+e)*f));
9
10     return 0;
11 }
```

Wrong Expectations

- Be careful when you copy and paste code.

```
113  #include <stdio.h>
114
115  int main(void)
116  {
117      printf("Hello World!!\n");
118      Printf("Hello World!!\n");
119      return 0;
120
121  }
```

```
C:\Users\z2017233\Desktop>gcc lecture2.c -o lecture2
lecture2.c: In function 'main':
lecture2.c:118:2: warning: implicit declaration of function 'Printf' [-Wimplicit-function-declaration]
  Printf(â?oHello World!!\nâ??);
  ^
lecture2.c:118:2: error: stray '\342' in program
lecture2.c:118:2: error: stray '\200' in program
lecture2.c:118:2: error: stray '\234' in program
lecture2.c:118:12: error: 'Hello' undeclared (first use in this function)
  Printf(â?oHello World!!\nâ??);
            ^
lecture2.c:118:12: note: each undeclared identifier is reported only once for each function it appears in
lecture2.c:118:18: error: expected ')' before 'World'
  Printf(â?oHello World!!\nâ??);
                  ^
lecture2.c:118:18: error: stray '\' in program
lecture2.c:118:18: error: stray '\342' in program
lecture2.c:118:18: error: stray '\200' in program
lecture2.c:118:18: error: stray '\235' in program

C:\Users\z2017233\Desktop>
```

Run-time Error Detection

- Array out-of-bound is not detected.

```
113  #include <stdio.h>
114
115  int main(void)
116  {
117      int arr[2];
118      arr[0] = 0;
119      arr[1] = 1;
120
121      int i = 0;
122      for(i = 0; i < 3; i++)
123      {
124          printf("%d\n", arr[i]);
125      }
126
127      return 0;
128
129  }
```

Overview

- Write a C Program
- Compile and Run a C Program
- Language Insecurities
- **Practice Hygienic Coding**

Hygienic Coding

- All variables, pointers and references are properly initialised at first and subsequent uses
- All input data, messages and output data should be validated
- Implementations of all algorithms should be validated
- Error handling
- Resource access are explicitly managed
- Use of comment statements
- Code layout and use of indenting
- Layout of braces “{ }” and block structures
- Statement complexity

Summary

- Write a C Program
- Compile and Run a C Program
- Language Insecurities
- Practice Hygienic Coding