The University of Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

Week 2 - Lecture 1

Writing a C program – fundamental

Overview

- Write a C Program
- Language Insecurities
- Practice Hygienic Coding

Write a C program - why use X2Go

- X2Go provides a pure C compiler
- X2Go provides an IDE
- Everyone has access to X2Go equally -- your submissions should be assessed by the same C compiler

Redundancy

- We always have multiple ways to express what we want to express with a language.

- Every programmer has their specific style in writing programs.

- There are some good habits in code writing in the sense of software engineering practice.

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.

```c
#include <stdio.h>

void myPrint(void);
int myReturn(void);

int main(void)
{
    int number = 0;

    myPrint();

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

    number = myReturn();

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

    return 0;

}

void myPrint(void)
{
    printf("Hello There !!\n");
}

int myReturn(void)
{
    return 5;
}
```

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

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

Libraries

```c
void main(void)
{

}
```

- Almost every program needs some libraries.
- Example libraries:
  - limits.h
  - math.h
  - stdio.h

```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;

}
```

- 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.

```c
#include <stdio.h>

int main(void)
{
    int counter;
    int number = 0;

    while(number < 1)
    {
        printf("Please enter a new number: ");
        scanf("%d", &number);

        counter = counter + 1;
    }

    if(counter > 1)
    {
        printf("User has entered %d new numbers\n", counter);
    }
    else
    {
        printf("User has entered %d new number\n", counter);
    }

    return 0;

}
```

```
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  }
```

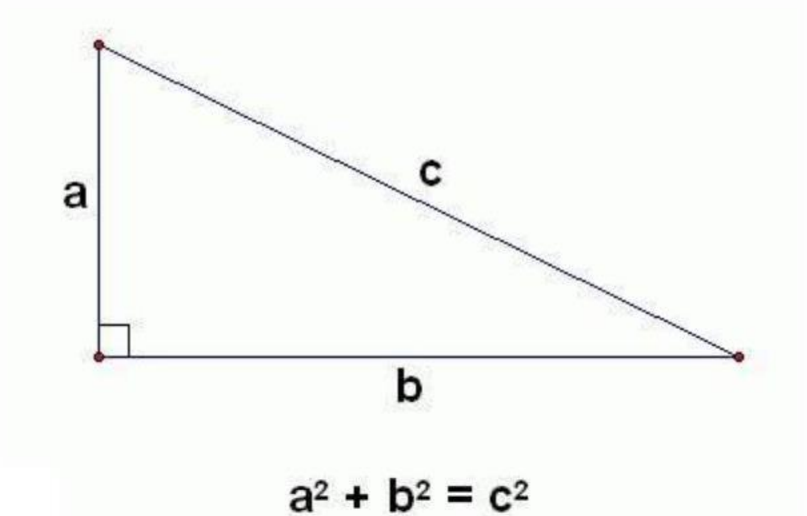
Example 1: Sum

```c
// Fig. 2.5: fig02_05.c
// Addition program.
#include <stdio.h>

// function main begins program execution
int main( void )
{
   int integer1; // first number to be entered by user
   int integer2; // second number to be entered by user
   int sum; // variable in which sum will be stored

   printf( "Enter first integer\n" ); // prompt
   scanf( "%d", &integer1 ); // read an integer

   printf( "Enter second integer\n" ); // prompt
   scanf( "%d", &integer2 ); // read an integer

   sum = integer1 + integer2; // assign total to sum

   printf( "Sum is %d\n", sum ); // print sum
} // end function main
```

```c
// Fig. 2.5: fig02_05.c
// Addition program.
#include <stdio.h>

// function main begins program execution
int main( void )
{
   int integer1; // first number to be entered by user
   int integer2; // second number to be entered by user
   int sum; // variable in which sum will be stored

   printf( "Enter first integer\n" ); // prompt
   scanf( "%d", &integer1 ); // read an integer

   printf( "Enter second integer\n" ); // prompt
   scanf( "%d", &integer2 ); // read an integer

   sum = integer1 + integer2; // assign total to sum

   printf( "Sum is %d\n", sum ); // print sum
} // end function main
```

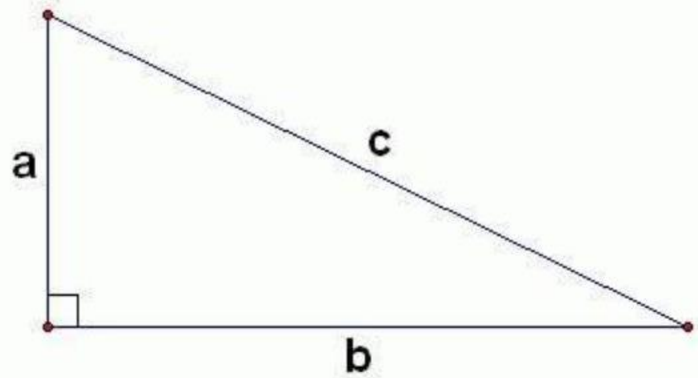
Example 2: Right-Angled Triangle

```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 );
    }

}
```



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

```c
    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 );
    }

}
```



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

Overview

- Write 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      float e = 2.5;
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.

```c
#include <stdio.h>

int main(void)
{
    printf("Hello World!!\n");
    Printf("Hello World!!\n");
    return 0;

}
```

```
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
- 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
- Language Insecurities
- Practice Hygienic Coding