Programming with C I

Fangtian Zhong
CSCI 112

Gianforte School of Computing
Norm Asbjornson College of Engineering
E-mail: fangtian.zhong@montana.edu

2024.01.29

Objectives

- To learn about functions and how to use them to write programs with separate modules.
- To understand the capabilities of some standard functions in C.
- To understand how control flows between function main and other functions.
- To learn how to pass information to functions using input arguments.
- To learn how to return a value from a function.

Top-Down Design

top-down design

- a problem solving method
- first, break a problem up into its major subproblems
- solve the subproblems to derive the solution to the original problem

House and Stick Figure

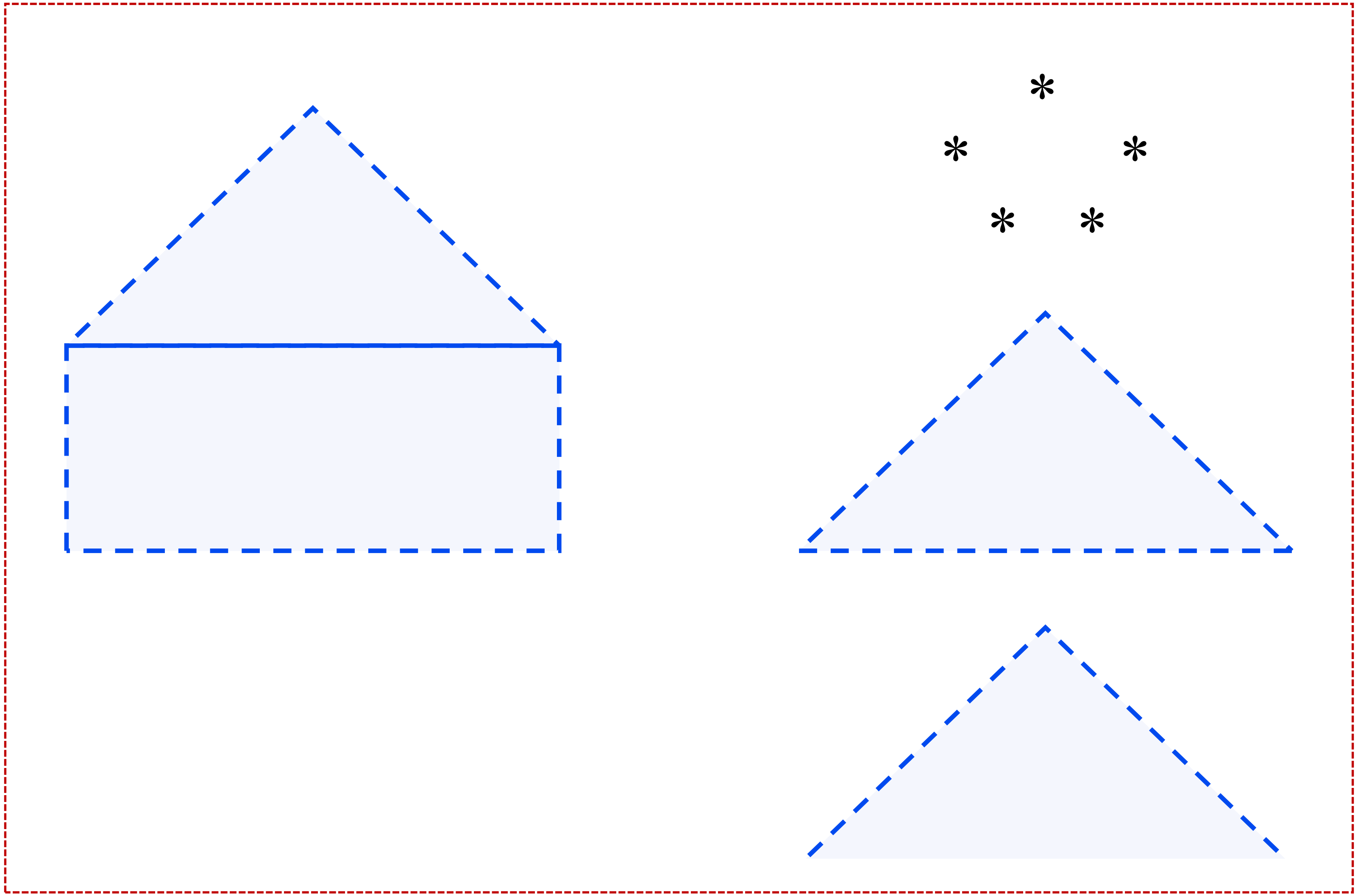
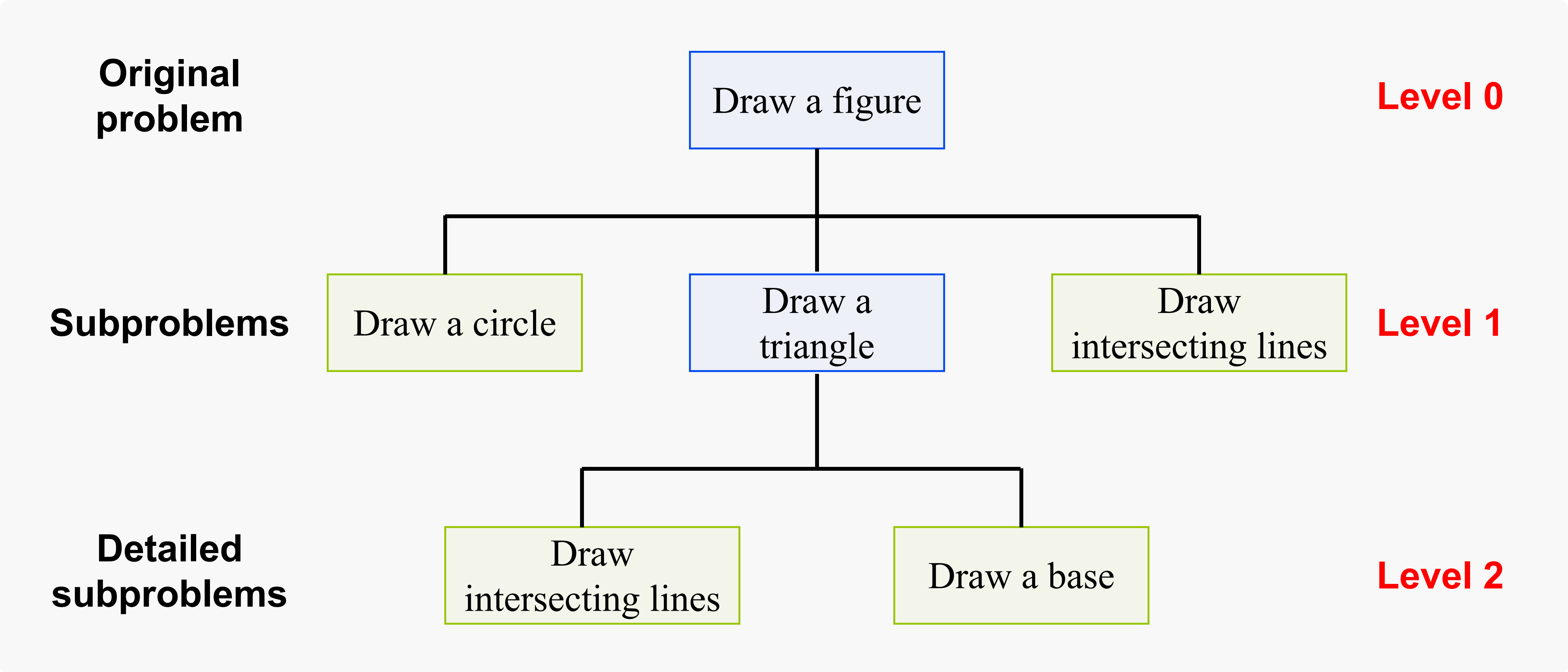


Figure Structure Chart for Drawing a Stick Figure



Functions Call Statement (Function Without Arguments)

Syntax

```
fname();
```

Example:

```
draw_circle();
```

Interpretation

- the function fname is called
- after fname has finished execution, the program statement that follows the function call will be executed

Figure Function Prototypes and Main Function for Stick Figure

```
/*
 * Draws a stick figure
 */

#include <stdio.h>                /* printf definition */

/* function prototypes */

void draw_circle(void);         /* Draws a circle                 */
void draw_intersect(void);      /* Draws intersecting lines       */
void draw_base(void);           /* Draws a base line              */
void draw_triangle(void);       /* Draws a triangle               */

int
main (void)
{
        /* Draw a circle. */
        draw_circle();

        /* Draw a triangle. */
        draw_triangle();

        /* Draw intersecting line. */
        draw_intersect();

        return (0);
}
```

Function Prototype (Function Without Arguments)

Syntax

```
ftype
fname(void);
```

Example:

```
void
draw_circle(void)
```

Interpretation

- the identifier fname is declared to be the name of a function
- the identifier ftype specifies the data type of the function result

Figure Function draw_circle

```
/*
 * Draws a circle
 */
void
draw_circle(void)
{
        printf(“    *   \n”);
        printf(“  *    *\n”);
        printf(“   *  * \n”);
}
```

Function Definitions (Function Without Arguments)

Syntax

```
ftype
fname(void)
{
            local declarations
            executable statements
}
```

Figure Function draw_triangle

```
/*
 * Draws a triangle
 */
void
draw_triangle(void)
{
        draw_intersect();
        draw_base();
}
```

Advantages of Using Function Subprogram

procedural abstraction

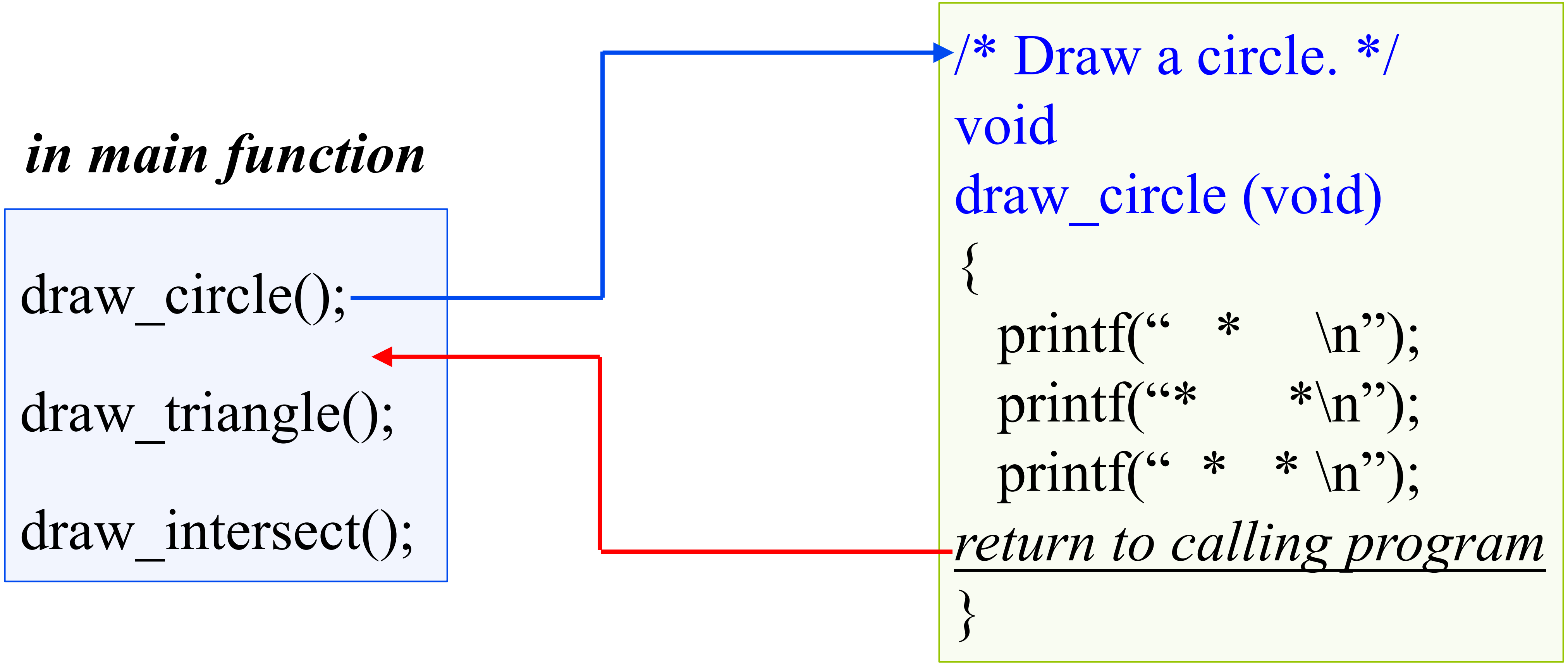
- a programming technique in which a main function consists of function calls and each function is implemented separately

reuse of function subprograms

- functions can be executed more than once in a program

Figure Flow of Control Between the main Function and a Function Subprogram

Computer memory



in main function

```
draw_circle();

draw_triangle();

draw_intersect();
```

```
/* Draw a circle. */
void
draw_circle (void)
{
   printf(“   *    \n”);
   printf(“*      *\n”);
   printf(“ *   * \n”);
return to calling program
}
```

Functions with Input Arguments

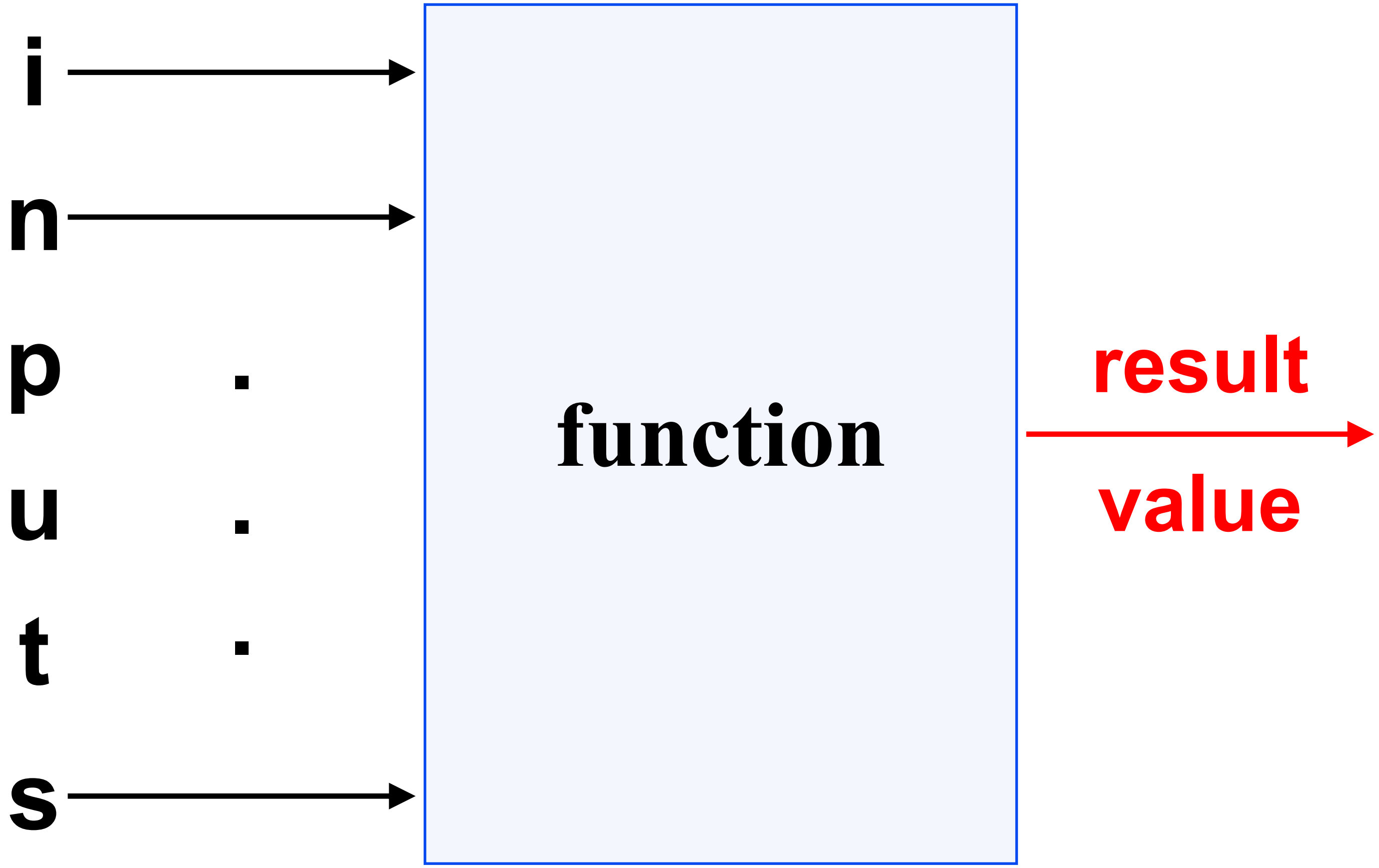
🏅 input argument

- arguments used to pass information into a function subprogram

🏅 output argument

- arguments used to return results to the calling function

Figure Function with Input Arguments and One Result



Functions with Multiple Arguments

Argument List Correspondence

- The number of actual arguments used in a call to a function must be the same as the number of formal parameters listed in the function prototype.
- Each actual argument must be of a data type that can be assigned to the corresponding format parameter with no unexpected loss of information.

Functions with Multiple Arguments

Argument List Correspondence

The order of arguments in the lists determines correspondence.

- The first actual argument corresponds to the first formal parameter.
- The second actual argument corresponds to the second form parameter.
- *etc*.

THE END

Fangtian Zhong
CSCI 112

Gianforte School of Computing
Norm Asbjornson College of Engineering
E-mail: fangtian.zhong@montana.edu

2024.01.29