



Lecture 09

Functions

CSE115: Computing Concepts

Introduction

- A function is a block of code which is used to perform a specific task.
- It can be written once and can be reused for a different program without having to rewrite that piece of code.
- Functions can be put in a library. If another program would like to use them, it will just need to include the appropriate header file at the beginning of the program and link to the correct library while compiling.

Introduction

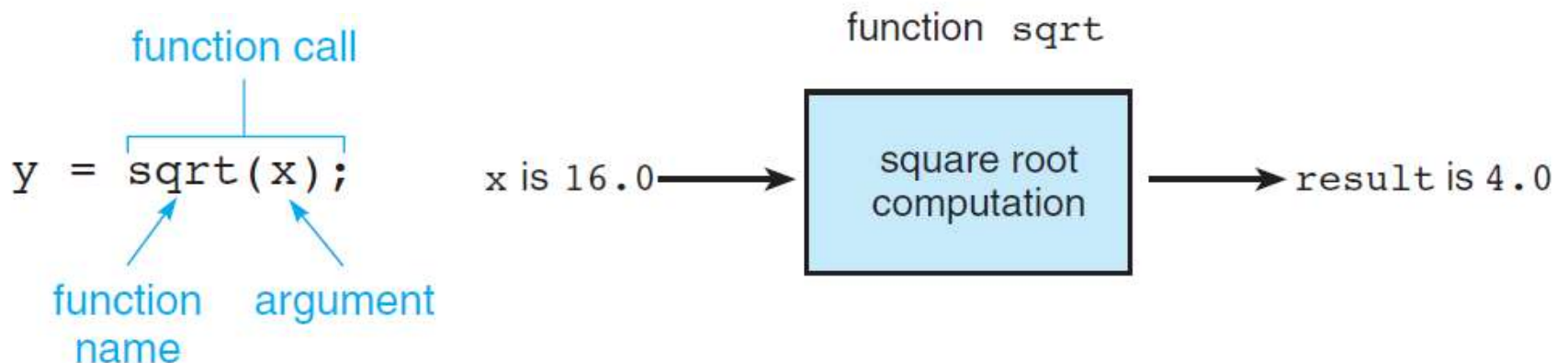
- Functions can be divided into two categories :
 - Predefined functions (standard functions)
 - Built-in functions provided by C that are used by programmers without having to write any code for them. i.e: printf(), scanf(), etc
 - User-Defined functions
 - Functions that are written by the programmers themselves to carry out various individual tasks.

Standard Functions

- Standard functions are functions that have been pre-defined by C and put into standard C libraries.
 - Example: `printf()`, `scanf()`, `pow()`, `ceil()`, `rand()`, etc.
- What we need to do to use them is to include the appropriate header files.
 - Example: `#include <stdio.h>`, `#include <math.h>`
- What contained in the header files are the prototypes of the standard functions. The function definitions (the body of the functions) has been compiled and put into a standard C library which will be linked by the compiler during compilation.

C Library Functions

- Avoid errors due to repetition of code
- Reusability of code
- C's standard math library defines a function named `sqrt` - performs the square root computation



C Library Functions

Function	Header	Purpose: Example	Argument(s)	Result
abs(x)	<stdlib.h>	Returns the absolute value of its integer argument: if x is -5 , abs(x) is 5	int	int
ceil(x)	<math.h>	Returns the smallest integral value that is not less than x : if x is 45.23 , ceil(x) is 46.0	double	double
cos(x)	<math.h>	Returns the cosine of angle x : if x is 0.0 , cos(x) is 1.0	double (radians)	double
pow(x,y)	<math.h>	Returns x^y . If x is negative, y must be integral: if x is 0.16 and y is 0.5 , pow(x,y) is 0.4	double, double	double
sqrt(x)	<math.h>	Returns the nonnegative square root of x (1x) for $x \geq 0.0$: if x is 2.25 , sqrt(x) is 1.5	double	double
log(x)	<math.h>	Returns the natural logarithm of x for $x > 0.0$: if x is 2.71828 , log(x) is 1.0	double	double
log10(x)	<math.h>	Returns the base-10 logarithm of x for $x > 0.0$: if x is 100.0 , log10(x) is 2.0	double	double

Character Handling Library

- Character handling library includes several function that perform useful tests and manipulation of character data.
- Each function receives a character, represented as an int or EOF, as an argument.
- When using functions from the character handling library, the header file **<ctype.h>** needs to be included.
- Characters in these functions are manipulated as integers (since a character is basically a 1 byte integer).

Functions in <ctype.h>

Prototype	Function Descriptions
int isdigit(int c)	Returns a true if value c is a digit, and 0 (false) otherwise.
int isalpha(int c)	Returns a true if value c is a letter, and 0 otherwise.
int isalnum(int c)	Returns a true if value c is a digit or a letter, and 0 otherwise.
int isxdigit(int c)	Returns a true value if c is a hexadecimal digit character, and 0 otherwise.
int islower(int c)	Returns a true value if c is a lowercase letter, and 0 otherwise.
int isupper(int c)	Returns a true value if c is an uppercase letter, and 0 otherwise.
int tolower(int c)	If c is an uppercase letter, tolower returns c as a lowercase letter. Otherwise, tolower returns the argument unchanged.
int toupper(int c)	If c is a lowercase letter, toupper returns c as an uppercase letter. Otherwise toupper returns the argument unchanged.
int isspace(int c)	Returns true if c is a white space character – newline ('\n'), space (' '), form feed ('\f'), carriage return ('\r'), horizontal tab ('\t') or vertical tab ('\v') – and 0 otherwise.
int iscntrl(int c)	Returns a true if c is a control character, and 0 otherwise.
int ispunct(int c)	Returns a true if c is a printing character other than a space, a digit or a letter, and 0 otherwise.
int isprint(int c)	Returns a true value if c is a printing character including space (' '), and 0 otherwise.
int isgraph(int c)	Returns a true value if c is a printing character other than space (' '), and 0 otherwise.

User Defined Functions

- A programmer can create his/her own function(s).
- It is easier to plan and write our program if we divide it into several functions instead of writing a long piece of code inside the main function.
- A function is **reusable** and therefore prevents us (programmers) from having to unnecessarily rewrite what we have written before.
- In order to write and use our own function, we need to do the following:
 - create a function prototype (declare the function)
 - define the function somewhere in the program (implementation)
 - call the function whenever it needs to be used

An Example

- Lets find the value of ${}^N C_R$, where the values of N and R will be provided by the user.
- FYI, ${}^N C_R = \frac{n!}{r! \times (n-r)!}$

An Example

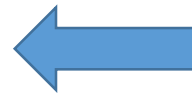
```
#include <stdio.h>

int main()
{
    int N, R;
    int factN = 1, factR = 1, factN_R = 1;
    int result;
    int i;
    printf("Enter N and R\n");
    scanf("%d%d", &N, &R);
    for(i=1; i<=N; i++)
        factN *= i;
    for(i=1; i<=R; i++)
        factR *= i;
    for(i=1; i<=(N-R); i++)
        factN_R *= i;
    result = factN / (factR * factN_R);
    printf("%d combination %d = %d", N, R, result);
    return 0;
}
```

An Example

```
#include <stdio.h>

int main()
{
    int N, R;
    int factN = 1, factR = 1, factN_R = 1;
    int result;
    int i;
    printf("Enter N and R\n");
    scanf("%d%d", &N, &R);
    for(i=1; i<=N; i++)
        factN *= i;
    for(i=1; i<=R; i++)
        factR *= i;
    for(i=1; i<=(N-R); i++)
        factN_R *= i;
    result = factN / (factR * factN_R);
    printf("%d combination %d = %d", N, R, result);
    return 0;
}
```



You are
implementing same
logic over and over

An Example

```
int factorial(int x)
{
    int fact = 1, i;
    for(i=1; i<=x; i++)
        fact *= i;
    return fact;
}
```

An Example

Function name



```
int factorial(int x)
{
    int fact = 1, i;
    for(i=1; i<=x; i++)
        fact *= i;
    return fact;
}
```

An Example


Type and name of
Input to the function (parameters)



```
int factorial(int x)
{
    int fact = 1, i;
    for(i=1; i<=x; i++)
        fact *= i;
    return fact;
}
```

An Example

Type of output value (return type) (void if no output)



```
int factorial(int x)
{
    int fact = 1, i;
    for(i=1; i<=x; i++)
        fact *= i;
    return fact;
}
```


An Example

```
int factorial(int x)
{
    int fact = 1, i;
    for(i=1; i<=x; i++)
        fact *= i;
    return fact;
}
```



Body of the function

An Example

```
#include <stdio.h>
int factorial(int x);
int main()
{
    int N, R;
    int result;
    int i;
    printf("Enter N and R\n");
    scanf("%d%d", &N, &R);
    result = factorial(N) / (factorial(R) * factorial(N-R));
    printf("%d combination %d = %d", N, R, result);
    return 0;
}
int factorial(int x)
{
    int fact = 1, i;
    for(i=1; i<=x; i++)
        fact *= i;
    return fact;
}
```

An Example

```
#include <stdio.h>
int factorial(int x);
int main()
```



Function prototype

```
{
    int N, R;
    int result;
    int i;
    printf("Enter N and R\n");
    scanf("%d%d", &N, &R);
    result = factorial(N) / (factorial(R) * factorial(N-R));
    printf("%d combination %d = %d", N, R, result);
    return 0;
}
```

Calling the function



```
int factorial(int x)
{
    int fact = 1, i;
    for(i=1; i<=x; i++)
        fact *= i;
    return fact;
}
```



Function implementation

Top Down Design

- Analyze the problem
- Break it into smaller sub-problems
- Solve the smaller problems as Functions
- Put them together in the **main() function**

A Simple Drawing Program

Problem:

- Write a program `DrawFigures.c` to draw a rocket ship (which is a triangle over a rectangle, over an inverted V), a male stick figure (a circle over a rectangle over an inverted V), and a female stick figure (a circle over a triangle over an inverted V)



rocket



male



female

Analysis:

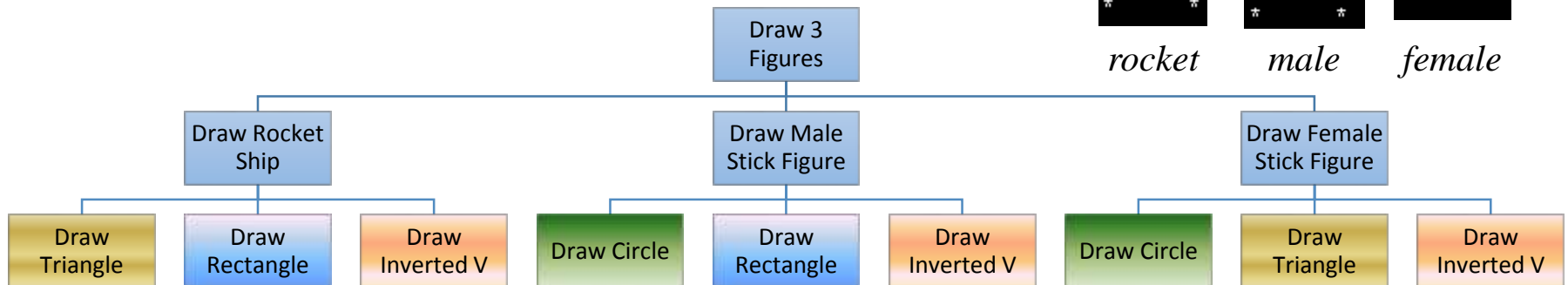
- No particular input needed, just draw the needed 3 figures
- There are common shapes shared by the 3 figures

Design:

- **Algorithm:**
 1. Draw Rocket ship
 2. Draw Male stick figure (below Rocket ship)
 3. Draw Female stick figure (below Male stick figure)

A Simple Drawing Program

Design (Structure Chart):



rocket



male



female

A Simple Drawing Program

```
#include <stdio.h>

void draw_rocket_ship();
void draw_male_stick_figure();
void draw_female_stick_figure();
void draw_circle();
void draw_rectangle();
void draw_intersect_line();

int main(void) {
    draw_rocket_ship();
    printf("\n\n");

    draw_male_stick_figure();
    printf("\n\n");

    draw_female_stick_figure();
    printf("\n\n");

    return 0;
}

void draw_rocket_ship()
{
    draw_triangle();
    draw_rectangle();
    draw_intersect_line();
}

void draw_male_stick_figure()
{
    draw_circle();
    draw_rectangle();
    draw_intersect_line();
}

void draw_female_stick_figure()
{
    draw_circle();
    draw_triangle();
    draw_intersect_line();
}
```

A Simple Drawing Program

```
void draw_circle() {  
    printf("  **  \n");  
    printf(" *    * \n");  
    printf(" *    * \n");  
    printf("  **  \n");  
}
```

```
void draw_rectangle() {  
    printf(" ***** \n");  
    printf(" *      * \n");  
    printf(" *      * \n");  
    printf(" *      * \n");  
    printf(" ***** \n");  
}
```

```
void draw_triangle() {  
    printf("   *   \n");  
    printf("  * *  \n");  
    printf(" *     * \n");  
    printf("*****\n");  
}
```

```
void draw_intersect_line()  
{  
    printf("   *   \n");  
    printf("  * *  \n");  
    printf(" *     * \n");  
    printf("*       *\n");  
}
```