

## User-Defined Functions

Er. Shiva K. Shrestha (HoD)

Department of Computer Engineering,

Khwopa College of Engineering



## Some Wonderful Numbers

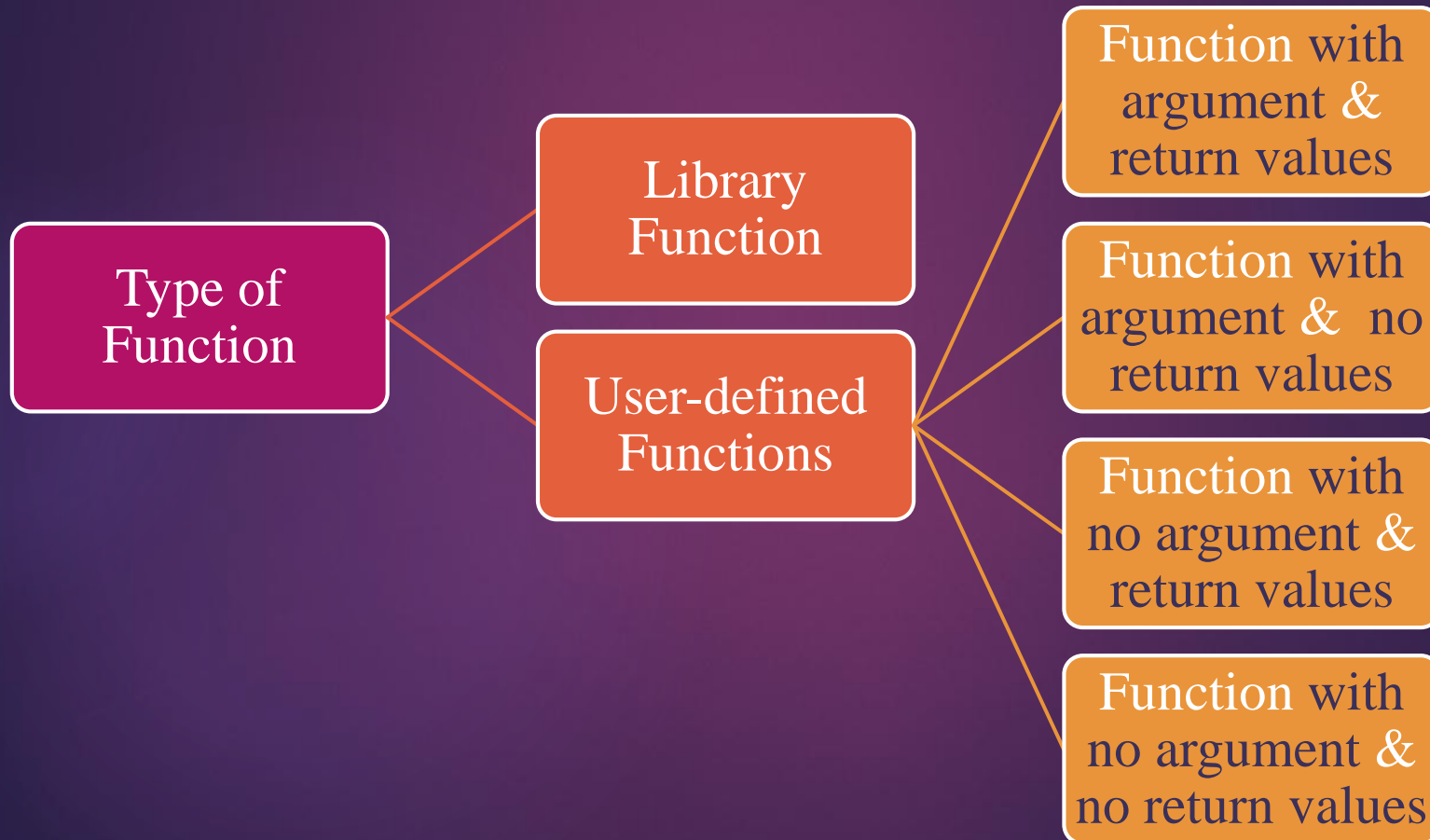
1. Armstrong Number
2. Palindrome Number
3. Prime Number
4. Twin Prime Numbers
5. Vesuvian Number
6. Triangular Number

## Examples

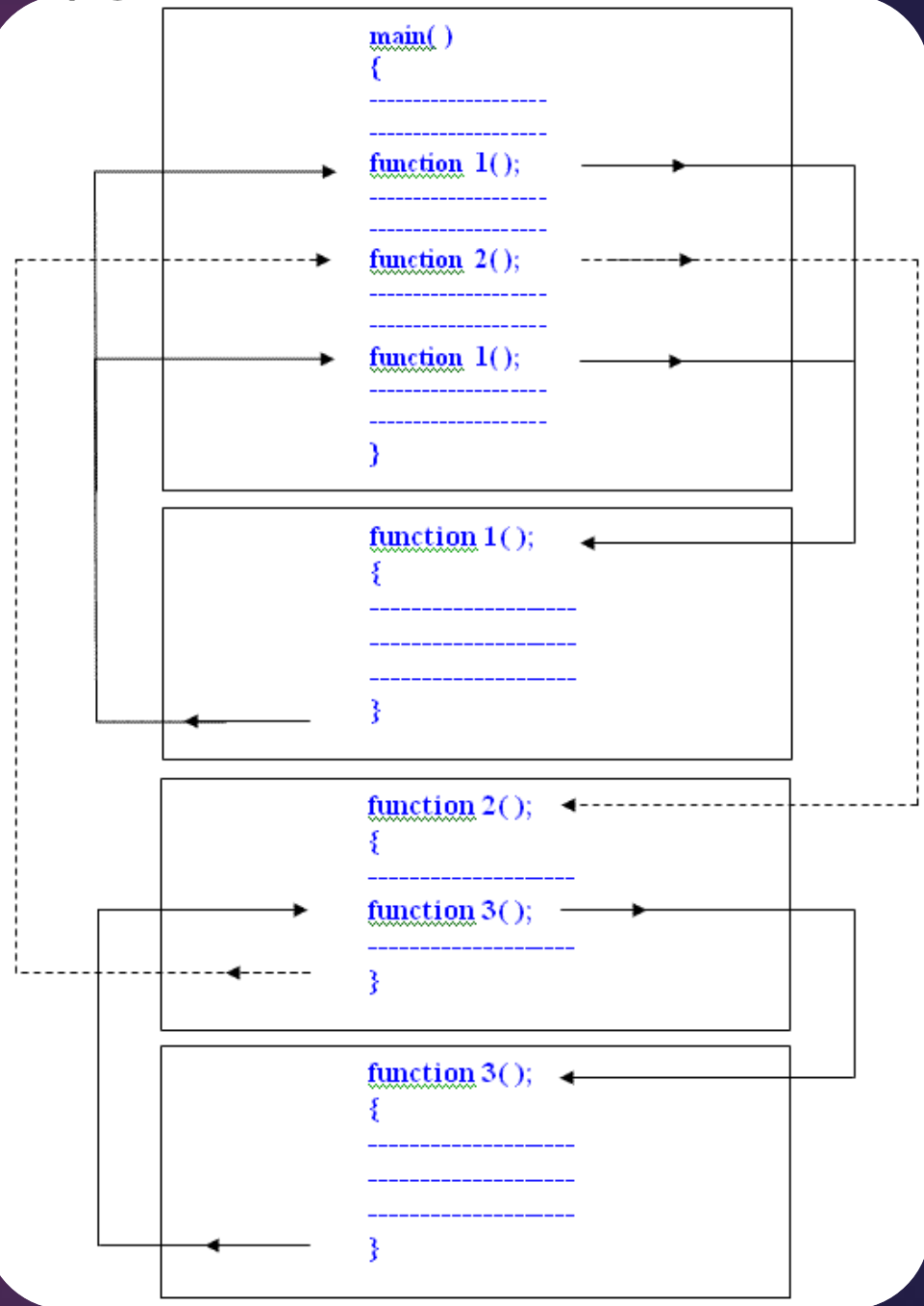
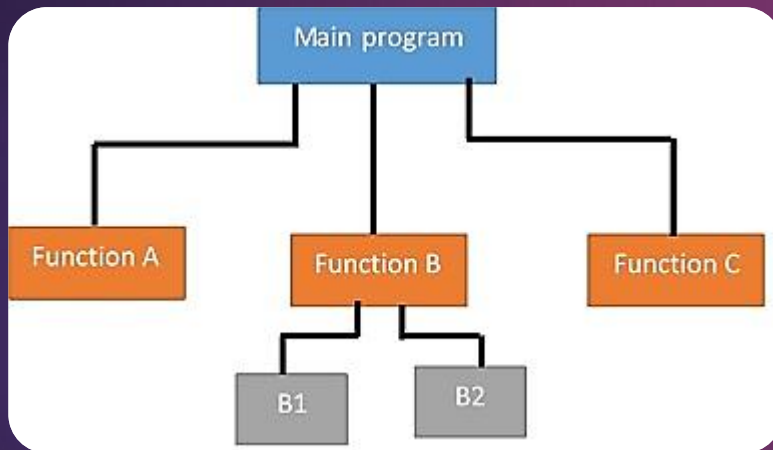
1.  $153 = 1^3 + 5^3 + 3^3$ , 0, 1, 370, 371, 1634:  $1^4 + 6^4 + 3^4 + 4^4$ ,
2. 11, 373, 12321,
3. 2, 3, 5, 7 ...
4. 29:  $29+2 = 31$  (Prime), 43:  $43-2 = 41$  (Prime)
5. If sum of 2 different pairs of square, 65:  $1^2 + 8^2$ , 85:  $2^2 + 9^2$ , 50, 125, 130, 145,
6. N is triangular if  $n = 1+2+3+\dots$ ,  $1=1$ ,  $3=1+2$ ,  $6=1+2+3$ , 10, 15,

# Function

- ▶ A function is a self-contained block of statements that performs a particular specified task.



# Control Flow in a multi-function program



# Advantages of Function

5

- ▶ Manageability
- ▶ Code Reusability
- ▶ Non-redundant Programming
- ▶ Logical Clarity
- ▶ Easy to divide the work among programmers

# Library Functions

## (Built-in Functions)

- ▶ Library functions in C language are inbuilt functions which are grouped together and placed in a common place called library.

Header File	Description
<u><a href="#">stdio.h</a></u>	For Standard input/output
<u><a href="#">conio.h</a></u>	For Console input/output
<u><a href="#">string.h</a></u>	All string related functions are defined
<u><a href="#">stdlib.h</a></u>	Contains general functions
<u><a href="#">math.h</a></u>	Contains all math related functions
<u><a href="#">time.h</a></u>	Contains time and clock related functions
<u><a href="#">ctype.h</a></u>	For Character handling functions

# User-defined Functions

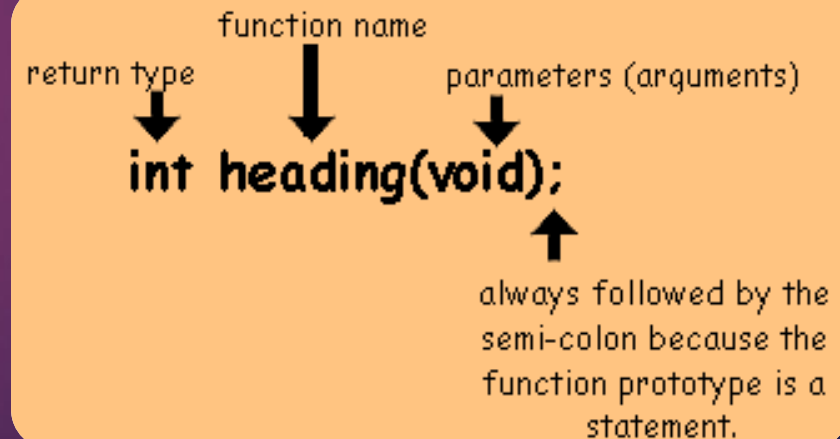
7

## Types of User-defined Functions

1. Function with argument & return values
2. Function with argument & no return values
3. Function with no argument & return values
4. Function with no argument & no return values

Er. Shiva K. Shrestha (HoD, Computer)  
10/3/2018

```
user_defined_functions.c x
1 int fxn1(int);
2 void fxn2(int);
3 int fxn3();
4 void fxn4();
```



# Components of Function

- ▶ Function Definition
  - ▶ Function Declarator & Function Body
- ▶ Function Declaration or Prototype
- ▶ Return Statement
- ▶ Accessing/Calling a function
- ▶ Function Parameters (Arguments)



# Function Definition

```
int find_max(int n1, int n2) {  
    int max;  
    if (n1 > n2){  
        max = n1;  
    }else{  
        max = n2;  
    }  
    return max;  
}
```

```
C6_1.C  
1  #include<stdio.h>  
2  #include<conio.h>  
3  int find_max(int, int);  
4  int main(){  
5      int large;  
6      large = find_max(8, 12);  
7      printf("Largest No.: %d", large);  
8      getch();  
9      return 0;  
10 }  
11  
12 /* function returning the max no. */  
13 int find_max(int n1, int n2) {  
14     /* local variable declaration */  
15     int max;  
16     if (n1 > n2){  
17         max = n1;  
18     }else{  
19         max = n2;  
20     }  
21     return max;  
22 }
```

# Return Statement

```
int find_max(int n1, int n2) {  
    int max;  
    if (n1 > n2){  
        max = n1;  
    }else{  
        max = n2;  
    }  
    return max;  
}
```

```
C6_1.C  
1  #include<stdio.h>  
2  #include<conio.h>  
3  int find_max(int, int);  
4  int main(){  
5      int large;  
6      large = find_max(8, 12);  
7      printf("Largest No.: %d",large);  
8      getch();  
9      return 0;  
10 }  
11  
12 /* function returning the max no. */  
13 int find_max(int n1, int n2) {  
14     /* local variable declaration */  
15     int max;  
16     if (n1 > n2){  
17         max = n1;  
18     }else{  
19         max = n2;  
20     }  
21     return max;  
22 }
```

# Function Prototype

11

```
C6_1.C
1  #include<stdio.h>
2  #include<conio.h>
3  int find_max(int, int);
4  int main(){
5      int large;
6      large = find_max(8, 12);
7      printf("Largest No.: %d", large);
8      getch();
9      return 0;
10 }
11
12 /* function returning the max no. */
13 int find_max(int n1, int n2) {
14     /* local variable declaration */
15     int max;
16     if (n1 > n2){
17         max = n1;
18     }else{
19         max = n2;
20     }
21     return max;
22 }
```

# Function Invocation

12

```
C6_1.C
1  #include<stdio.h>
2  #include<conio.h>
3  int find_max(int, int);
4  int main(){
5      int large;
6      large = find_max(8, 12);
7      printf("Largest No.: %d", large);
8      getch();
9      return 0;
10 }
11
12 /* function returning the max no. */
13 int find_max(int n1, int n2) {
14     /* local variable declaration */
15     int max;
16     if (n1 > n2){
17         max = n1;
18     }else{
19         max = n2;
20     }
21     return max;
22 }
```

# Category of User-defined Functions according to Return Values & Arguments

1. Function with argument & return values
2. Function with argument & no return values
3. Function with no argument & return values
4. Function with no argument & no return values

# 1. Function with argument & return values

14

Er. Shiva K. Shrestha (HoD, Computer Department)  
10/3/2018

```
1  /* 1. Function with argument
2  & return values */
3  #include<stdio.h>
4  #include<conio.h>
5  int findSquare(int);
6  int main(){
7      int sq;
8      sq = findSquare(5);
9      printf("Square = %d",sq);
10     getch();
11     return 0;
12 }
13
14 int findSquare(int n) {
15     return (n*n);
16 }
```

## 2. Function with argument & no return values

15

Er. Shiva K. Shrestha (HoD, Computer Department)  
10/3/2018

```
1  /* 1. Function with argument
2  & no return values */
3  #include<stdio.h>
4  #include<conio.h>
5  void findSquare(int);
6  int main(){
7      findSquare(5);
8      getch();
9      return 0;
10 }
11
12 void findSquare(int n) {
13     printf("Square = %d", (n*n));
14 }
```

# 3. Function with no argument & return values

16

Er. Shiva K. Shrestha (HoD, Computer Department)  
10/3/2018

```
C6_4.C x
1  /* 1. Function with no argument
2  & with return values */
3  #include<stdio.h>
4  #include<conio.h>
5  int findSquare();
6  int main(){
7      int sq;
8      sq = findSquare();
9      printf("Square = %d",sq);
10     getch();
11     return 0;
12 }
13
14 int findSquare() {
15     int n = 7;
16     return (n*n);
17 }
```



## 4. Function with no argument & no return values

17

Er. Shiva K. Shrestha (HoD, Computer Department)  
10/3/2018

```
1  /* 1. Function with no argument
2  & with no return values */
3  #include<stdio.h>
4  #include<conio.h>
5  void findSquare();
6  void main(){
7      findSquare();
8      getch();
9  }
10
11 void findSquare() {
12     int n = 3;
13     printf("Square = %d", (n*n));
14 }
```

# Task

18

- ▶ Write four separate programs to add two entered integers using all type of user-defined functions.
- ▶ Expected Output:

```
Enter a: 5  
Enter b: 7  
Sum = 12
```

# Types of Function Call

19

1. Function call by value (or Pass arguments by value)
2. Function call by reference (or Pass arguments by address)

# Call By Value

20

```
C6_6.C
1  /* Call By Value*/
2  #include<stdio.h>
3  #include<conio.h>
4  void swap(int ,int);
5  int main(){
6      int x = 5;
7      int y = 7;
8      swap(x, y);
9      printf("x = %d\ty = %d", x, y);
10     getch();
11     return 0;
12 }
13 void swap(int a,int b){
14     int t;
15     t = a;
16     a = b;
17     b = t;
18     printf("a = %d\tb = %d\n", a, b);
19 }
```



C:\Users\ErSKS\Google Drive

```
a = 7    b = 5
x = 5    y = 7
```

# Call By Reference

21

Er. Shiva K. Shrestha  
10/3/2018

C6\_7.C

```
1  /* Call By Reference*/
2  #include<stdio.h>
3  #include<conio.h>
4  void swap(int *, int *);
5  int main(){
6      int m=5;
7      int n=7;
8      swap(&m, &n);
9      printf("The value of m and n ");
10     printf("after swapping is %d & %d", m, n);
11     getch();
12     return 0;
13 }
14 void swap(int *a, int *b){
15     int t;
16     t = *a;
17     *a = *b;
18     *b = t;
19 }
```

C:\Users\ErSKS\Google Drive (c.khwopa@gmail.com)\C\_Codes\C6\_7.exe

The value of m and n after swapping is 7 & 5

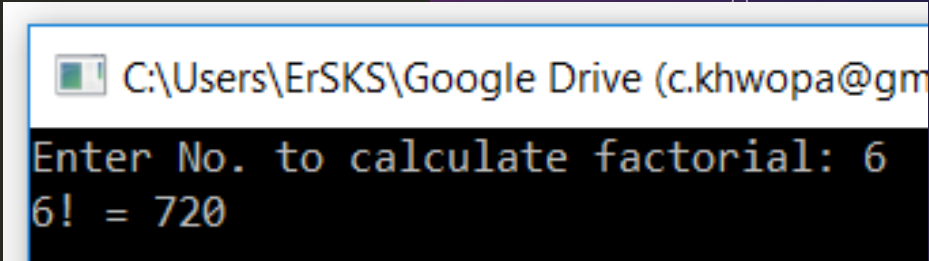
Computer Department)

# Recursive Functions

22

Er. Shiva K. Shrestha (HoD, Comp)  
10/3/2018

```
C6_8.C x
1  #include<stdio.h>
2  #include<conio.h>
3  long fact(int);
4  int main(){
5      int n;
6      long y;
7      printf("Enter No. to calculate factorial: ");
8      scanf("%d",&n);
9      y = fact(n);
10     printf("%d! = %ld", n, y);
11     getch();
12     return 0;
13 }
14 long fact(int z){
15     long value;
16     if(z==1){
17         return 1;
18     }else{
19         value = z*fact(z-1); //recursion
20     }
21     return value;
22 }
```



C:\Users\ErSKS\Google Drive (c.khwopa@gm)

Enter No. to calculate factorial: 6  
6! = 720

# Macros

23

Er. Shiva K. Shrestha (HoD, Computer Department)  
10/3/2018

```
C6_11.C
1  #include<stdio.h>
2  #include<conio.h>
3  #define area l*b
4  int main(){
5      int l,b;
6      printf("Enter l and b:\n");
7      scanf("%d%d",&l, &b);
8      printf("Area = %d",area);
9      getch();
10     return 0;
11 }
```

```
C:\Users\ErSKS\
Enter l and b:
5 4
Area = 20
```

# Macros with Argument

24

Er. Shiva K. Shrestha (HoD, Computer Department)  
10/3/2018

```
C6_12.C
1  #include<stdio.h>
2  #include<conio.h>
3  #define area(x) (3.14*x*x)
4  int main(){
5      float r1=6.25, r2=2.5, a;
6      a=area(r1);
7      printf("Area of Circle = %f",a);
8      a=area(r2);
9      printf("\nArea of Circle = %f",a);
10     getch();
11     return 0;
12 }
```

C:\Users\ErSKS\Google Drive (c.k

```
Area of Circle = 122.656250
Area of Circle = 19.625000
```



# Scope, Visibility & Lifetime of Variables

25

1. Automatic Variables,
2. External Variables,
3. Static Variables,
4. Register Variables

# Automatic Variables

26

- ▶ Automatic variable is a local variable which is allocated and de-allocated automatically when program flow enters and leaves the variable's scope. The keyword **auto** is used to declare automatic variables explicitly.

```
main(){  
    auto int number;  
    ...  
}
```

# External Variables

- ▶ It is possible to define variables that are external to all functions, that is, variables that can be accessed by name by any function. External variables are both **active** and **alive** throughout the entire program so these are also known as **global variables**.

```
int global_variable;  
  
void main(void) {  
    global_variable = 1;  
    // Statements  
}
```

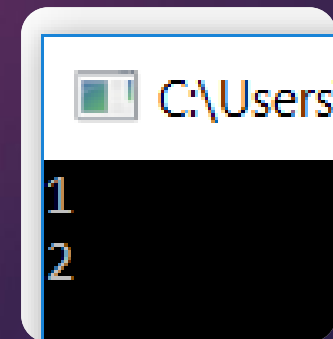
# Static Variables

- ▶ Static variables have a property of preserving their value even after they are out of their scope. Hence, static variables preserve their previous value in their previous scope and are not initialized again in the new scope.

Syntax:

- ▶ `static data_type var_name = var_value;`
- ▶ E.g. `static int a = 5;`

```
C6_13.C x
1  #include<stdio.h>
2  #include<conio.h>
3  int fun(){
4      static int count = 0;
5      count++;
6      return count;
7  }
8
9  int main(){
10     printf("%d\n", fun());
11     printf("%d\n", fun());
12     getch();
13     return 0;
14 }
```



# Register Variables

- ▶ Registers are faster than memory to access, so the variables which are most frequently used in a C program can be put in registers using register keyword. Register variables, are a special case of automatic variables, are kept by Compiler in one of the machine's registers instead of keeping in the memory for faster execution.

Syntax:

- ▶ `register data_type variable_name = value;`
- ▶ E.g. `register int count = 0;`

# Q/A?

30

## Thank You!

Er. Shiva K. Shrestha

[computer.khwopa@gmail.com](mailto:computer.khwopa@gmail.com)

Er. Shiva K. Shrestha (HoD, Computer Department)  
10/3/2018