Functions

- Function: a collection of statements to perform a task
- Construct a program from smaller pieces or components.
- Each piece is more manageable than the original program.
- Functions are invoked by function calls.

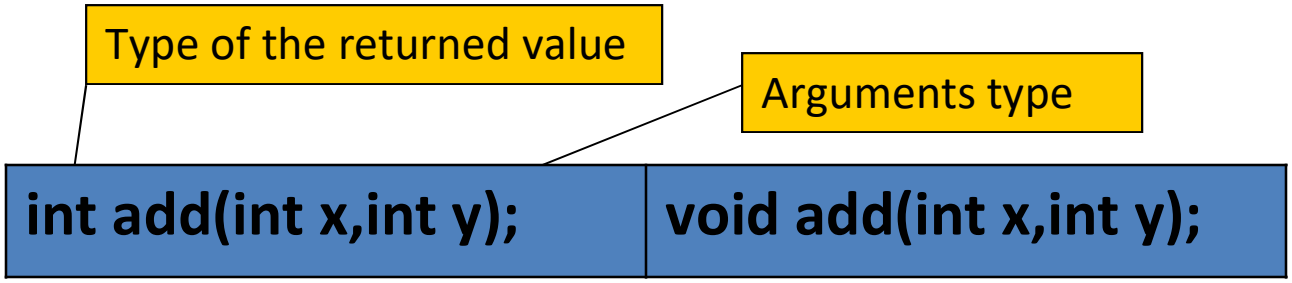
Functions

- Functions written once and can be used many.
- Function calls can be exist inside another function.
- Value returning functions return a value.
- Void functions do not return a value.

Function definition

- Function Prototype:- tells compiler the arguments type and the type of the function

Example:-



- Function Body:-

Statements that construct the function.

Example:-

int add(int x,int y){ **return x+y;** **}**	**void add(int x,int y){** **printf("%d",x+y);** **}**

Function definition

- Function Call:-

  The statement that invokes the function and pass arguments to the function.

Value returning functions can be called through	
Output statement	printf("%d",add(x,y));
Assignment statement	z= add(x,y);
Comparison statement	z==add(x,y);
Part of expression	z=add(x,y)+add(m,w)

Function definition

- Function Call:-

  The statement that invokes the function and pass arguments to the function.

void functions	
Can be called only by writing the name followed by the parameters list	**add(x,y);**

Example

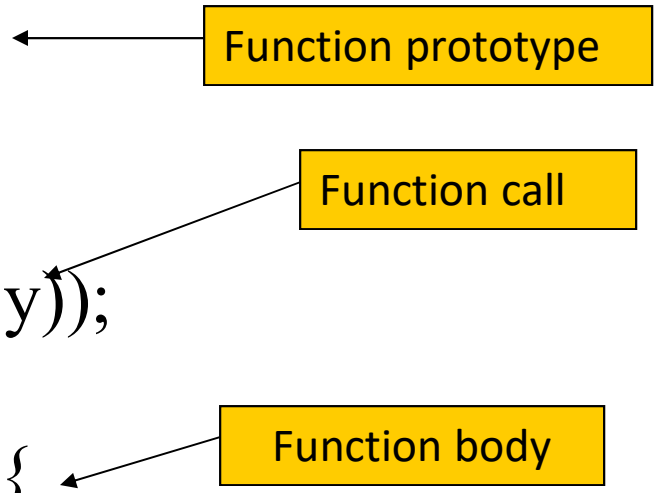
By Using functions, write a C++ program in order to add two integer numbers.

A. By using a function that returns an integer value.

B. By using a function that does not return a value.

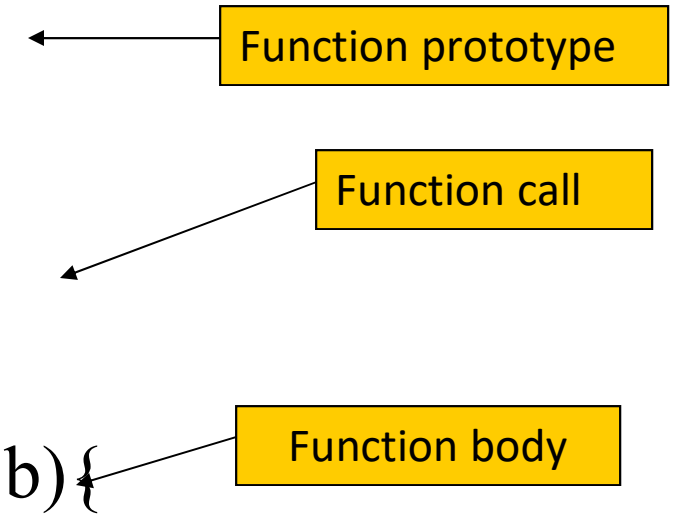
Solution part-A

```
#include<stdio.h>
int add(int, int);
void main(){
int x=5,y=7;
printf("%d",add(x,y));
}
int add(int a, int b){
return a+b;
}
```



Solution part-B

```
#include<stdio.h>
void add(int, int);
void main(){
int x=5,y=7;
add(x,y);
}
void add(int a, int b){
printf("%d",a+b);
}
```



Function prototype

- From the previous example, the syntax of function prototype is:

return_type name(arg1_type,arg2_type,…..);

- It may contain dummy variable names, for example:-

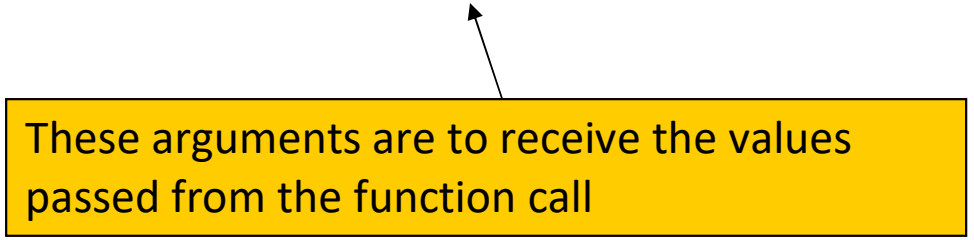
  int add(int w,int z);

- The absence of function prototype generates compilation error. Because the function prototype tells the compiler about the function return type, function name, and about the arguments type.
- If the function does not return results, its return type is *void*

Function body

- The syntax of the function body is

```
return_type name(arg1_type arg1,...argn_type
   argn){
statemets
}
```



- Return type is void if the function does not return a result.

Function call

To call a function, use the function name followed by () and ;

add(2,3);

When called, program executes the body of the called function

After the function terminates, execution resumes in the calling function at point of call.

Function call

- main can call any number of functions
- Functions can call other functions
- Compiler must know the following about a function before it is called:
  - name
  - return type
  - number of parameters
  - data type of each parameter

Example

Write a C++ program that reads two positive integer numbers x,y and passes the of x,y to a function that returns x^y

Solution

```
#include<stdio.h>
int power(int,int);
void main(){
int x,y; scanf("%d%d",&x,&y);
printf("%d",power(x,y));
}
int power(int a,int b){int z=1;
for(int i=1;i<=b;i++)z*=a;  return z;}
```

Example

By using functions write a C++ program that reads two numbers from the keyboard and the symbol of the operation that will be performed on these two numbers. Such that to perform the addition operation on 3.5 and 4.7, the program must read 3.5 + 4.7

Where + is the symbol for addition, - is the symbol for subtraction, * is the symbol for multiplication , / is the symbol for division

Solution

How many tasks do we have?

1) **Reading the numbers and the symbol**
2) **Deciding which operation will be performed on the input numbers.**
3) **Addition.**
4) **Subtraction.**
5) **Multiplication.**
6) **Division.**
7) **Printing the result**

Solution- prototypes and heders

```
#include<stdio.h>
double add(double, double);
double subtract(double, double);
double mult(double, double);
double divide(double, double);
```

Solution-main

```
void main(){
double x,y,z;
char op;  scanf("%lf%c%lf",&x,&op,&y);
switch(op){
case '+': z= add(x,y);break;
case '-': z = subtract(x,y);break;
case '*': z= mult(x,y);break;
case '/' :z = divide(x,y);break;
}
printf("%lf\n",z);
}
```

Solution- bodies

```
double add(double a,double b){
return  a+b;}
double subtract(double a,double b){
return  a-b;}
double mult(double a,double b){
return  a*b;}
double divide(double a,double b){
return  a/b;}
```

The Scope and Lifetime of a Variable

- **Variable's scope -** indicates which portions of a program can use the variable (can be either local or global)

- **Local variables -** variables declared within a function, and those that appear in a function's parameter List

Example

Write a C++ program to calculate e^x by using the following series

$$e^x = 1 + \sum_{i=1}^{100} \frac{x^i}{i!}$$

Note :You can use the power function pow(x,y) from math .h

Solution

- For each term in the series we need to calculate the factorial of i.
- So we have an operation that will be performed many times.
- The best way is to define this operation in a function that can be called many times.

Solution-function part

```
#include<iostream.h>
#include<math.h>
double factorial(double a){
double z =1;
for(int i=1;i<=a;i++)z= z*i;
return z;
}
```

Solution-main part

```
void main(){
double result=1,x;  cin>>x;
for(int i=1;i<=100;i++)
   result+= pow(x,i)/factorial(i);
printf("%f\n",result);
}
```

Example

Given:

```
double f(int x){return 3;}
double g(){double a =f(7)+2; cout<<a; return a;}
void main(){
double x=g()-f(1);printf("%f",x);}
}
```

What is printed??

Answer : 5 2

Example

What is the output of the following code?

```
int add(int a,int b){return a+b; printf(”hello”);}
void main(){
printf(“%d”,add(2,3));
}
```

Answer : 5

Statements after the return statement will never be executed.

Example

What is the output of the following code?

```
int add(int a,int b){ return a+b;
                      return a-b;
                      return a*b;}
void main(){
printf("%d",add(2,3));
}
```

Answer : 5

Only one return statement will be executed

Example

What is the output of the following code?

```
int add(int a,int b){ return a+b; }
void main(){
printf("%d",add('A','B'));
printf("%d",add(2.3,4.6));
}
```

Answer : 131 6

Data type conversion is performed when passing the values of the parameters

Example

What is the output of the following code?

```
char add(int a,int b){ return a+b; }
void main(){
printf("%c",add(32,33));
}
```

Answer : A

Data type conversion is performed on the returned result.

Example

What is the output of the following code?

```
int add(int a,int b){ return a+b; }
void main(){
int x=2,y=3;
printf("%d",add(x++,++y));
}
```

Answer : 6

Post increment and pre increment can be performed when passing the values of the parameters.

Example

What is the output of the following code?

```
int mul(int a,int b){ return ++a*(++b); }
void main(){
int x=2,y=3;
printf("result=%d\tx=%d\ty=%d",mul(x,y),x,y);
}
```

Answer : result= 12 x=2 y=3

when passing the values of the parameters, the modifications on the parameters inside the function do not affect any other variable outside the function scope.

Example

Write a function called multiple that determines for a pair of integers whether the second integer is a multiple of the first. The function should take two integer arguments and return true if the second is a multiple of the first, false otherwise. Use this function in a program that inputs 50 pairs of integers.

Solution

```
#include<stdio.h>
bool multiple(int a,int b){
if(b%a==0)return true;
else return false;}
void main(){
int x,y;
for(int i=1;i<=50;i++){
scanf("%d%d",&x,&y);
printf("%d"multiple(x,y));}
}
```

Example

Write a C++ function that takes the time as three integer arguments(hours, minutes, seconds) and returns the number of seconds since the last midnight. And use this function to calculate the amount of time in seconds between two times.

Hint: The time at every midnight is

00: for hours 00:minutes00:seconds

Solution

```
#include<stdio.h>
int time1(int h,int m,int s){
return h*3600+m*60+s; }
void main(){
printf("%d",time1(14,30,20)-time1(8,15,40));
}
```

Example

An integer is said to be perfect number if the sum of its factors, including 1 (but not the number itself), is equal to the number. For example 6 is perfect because 1+2+3=6.

Write a function called perfect that takes an integer parameter and determine whether the parameter number is a perfect number.

Use this function to print the perfect numbers from 1 to 100.

Solution

```
#include<stdio.h>
bool perfect(int a){
int sum=0;
for(int i=1;i<a;i++)
        if(a%i==0)sum+=i;
if(a==sum)return true;
else return false;  }
void main(){
for(int k=1;k<=100;k++)
   if(perfect(k)==true)printf("%d\n",k);
}
```

Example

Write a C++ function that takes an integer value and returns the number with its digits reversed.

Ex: given the number 7631 the function returns 1367.

Solution

```
#include<math.h>
int reverse(int a){
int x=a,r,count =0,sum=0;//first count the digits
while(x>0){x/=10;count++;}
while(a>0){r=a%10;a/=10;
sum+=r*pow(10,count-1);count--;}
}
```

Example

Write a C++ function called **gcd** that takes two integer parameters and returns the greatest common divisor of these two integers.

Where the greatest common divisor of two integer numbers is the largest integer that evenly divides each of the numbers

Solution

```
int gcd(int a,int b){
for(int i=a;i>=1;i--)
   if(a%i==0&&b%i==0)break;
return i;
}
```

Example

Write a C++ function called **grade** that takes the student's average as integer parameter and returns

- 'A' if the student average between 90-100
- 'B' if the student average between 80-90
- 'C' if the student average between 70-80
- 'D' if the student average between 60-70

Solution

```
char grade(int x){
if(x>=90&&x<=100)retrun ‘A’;
else if(x>=80&&x<=90)retrun ‘B’;
else if(x>=70&&x<=80)retrun ‘C’;
else if(x>=60&&x<=70)retrun ‘D’;
}
```

Example

Write a C++ function called **hypotenuse** that takes the two sides of a right triangle as two double arguments and returns the hypotenuse length of that triangle.

Solution

```
double hypotenuse(double a, double b){
double z=sqrt(pow(a,2)+pow(b,2));
return z;
}
```

Example

Write a C++ function that takes a number as an argument and returns the absolute value of that number.

solution

```
double absolute(double x){
if(x<0)return -1*x;
else return x;
}
```