SNS COLLEGE OF TECHNOLOGY

Coimbatore-35.

An Autonomous Institution

COURSE NAME : 23ITT101 PROGRAMMING IN C & DATA STRUCTURES I YEAR/

II SEMESTER

UNIT-II C DECISION STATEMENTS & FUNCTIONS

Topic: Functions

M.MOHANA PRIYA
Assistant Professor
Department of Artificial Intelligence and Machine Learning

FUNCTIONS

INTRODUCTION

➢ The strengths of C language is C functions.

➢ They are easy to define and use.

➢ We have used functions in every program that we have discussed so far.

➢ However, they have been primarily limited to the three functions, namely

main, printf, and scanf.

➢ C functions can be classified into two categories, namely, library functions and userdefined functions.

➢ main is an example of user-defined functions.

➢ printf and scanf belong to the category of library functions.

➢ The main distinction between these two categories is that library functions are not required to be written by us.

➢ Whereas a user-defined function has to be developed by the user at the time of writing a program.

➢ However, a user-defined function can later become a part of the C program library.

➢ In fact, this is one of the strengths of C language.

DEFINITION OF FUNCTIONS

* A function definition, also known as function implementation shall include the following elements:

1. function name;
2. function type;
3. list of parameters;
4. local variable declarations;
5. function statements; and
6. a return statement.

* All the six elements are grouped into two parts, namely,
* function header (First three elements); and
* function body (Second three elements).

DEFINITION OF FUNCTIONS

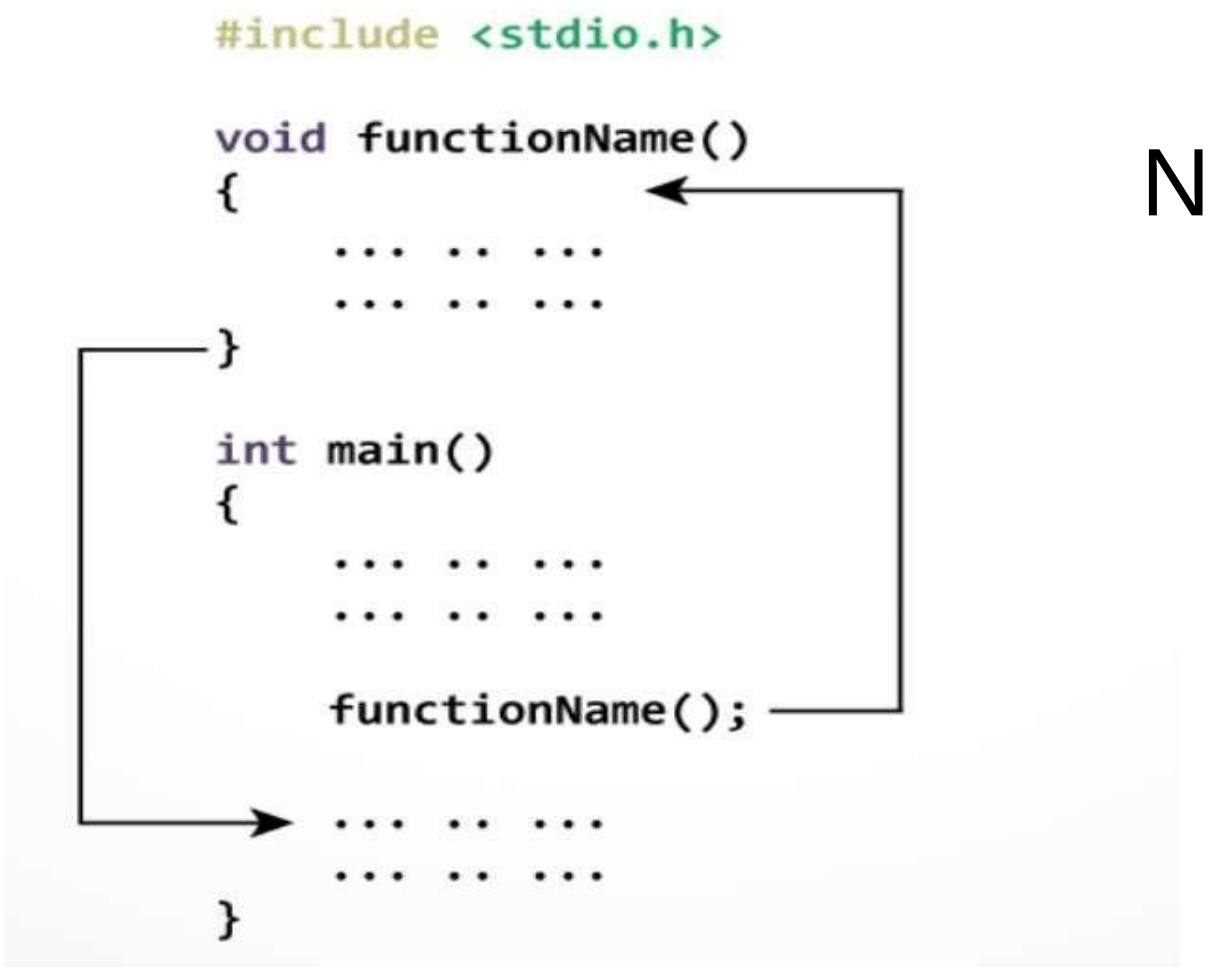
➢ A general format of a function definition to implement these two parts is given below:

```
function_type function_name(parameter list)
{
local variable declaration;
executable statement1;
executable statement2;
. . . . .
. . . . .
return statement;
}
```

• The first line function_type function_name(parameter list) is known as the function header and the statements within the opening and closing braces constitute the function body, which is a compound statement.

Functions

How function works in C programming?



Note, function names are identifiers and should be unique.

Functions

Function definition

Function definition contains the block of code to perform a specific task. In our example, adding two numbers and returning it.

Syntax of function definition

```
returnType functionName(type1 argument1, type2 argument2, ...)
{
    //body of the function
}
```

When a function is called, the control of the program is transferred to the function definition. And, the compiler starts executing the codes inside the body of a function.

Functions

Syntax of function prototype

```
returnType functionName(type1 argument1, type2 argument2, ...);
```

In the above example, `int addNumbers(int a, int b);` is the function prototype which provides the following information to the compiler:

1. name of the function is `addNumbers()`
2. return type of the function is `int`
3. two arguments of type `int` are passed to the function

The function prototype is not needed if the user-defined function is defined before the `main()` function.

Functions

Types of function

There are two types of function in C programming:

- Standard library functions
- User-defined functions

Functions

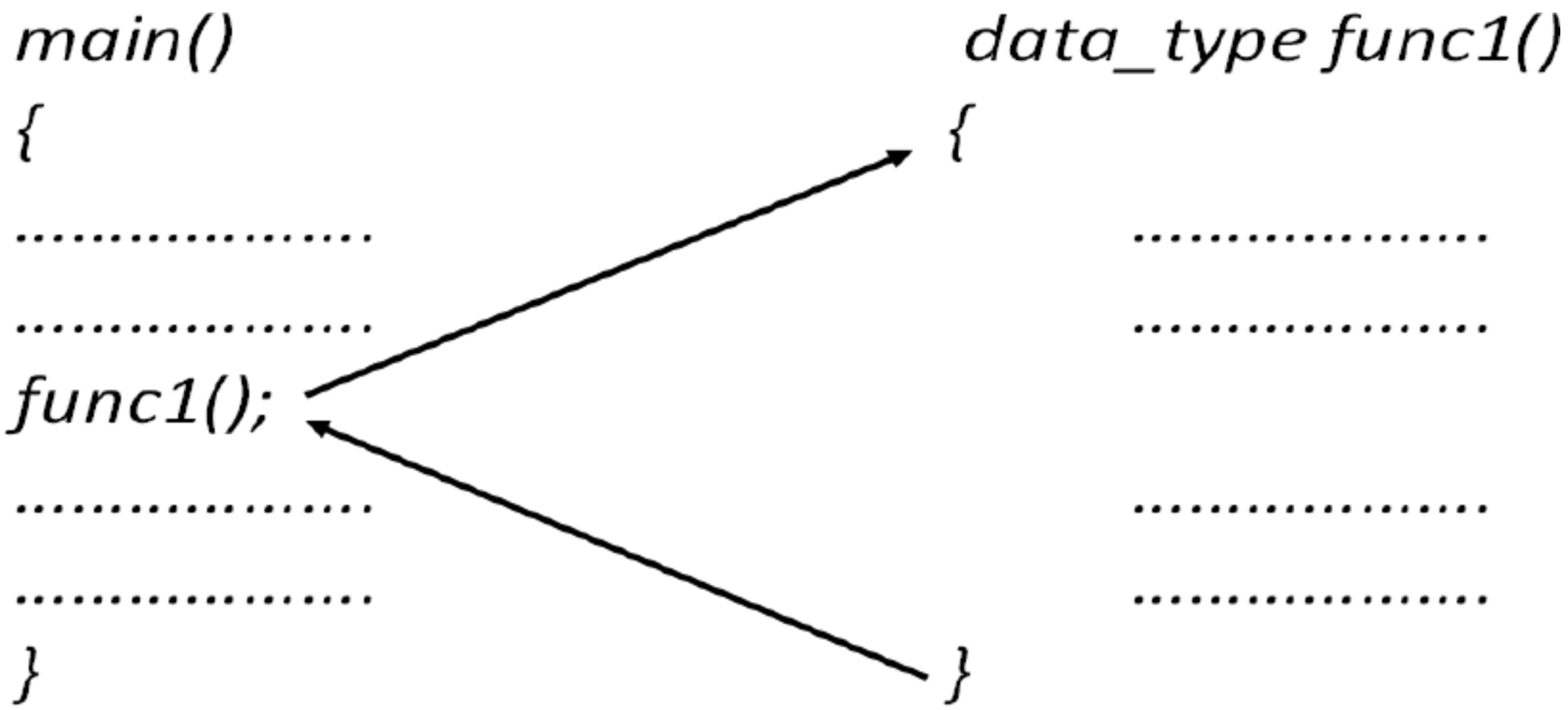
Standard library functions

The standard library functions are built-in functions in C programming.

These functions are defined in header files. For example,

- The `printf()` is a standard library function to send formatted output to the screen (display output on the screen). This function is defined in the `stdio.h` header file. Hence, to use the `printf()` function, we need to include the `stdio.h` header file using `#include <stdio.h>`.
- The `sqrt()` function calculates the square root of a number. The function is defined in the `math.h` header file.

Functions



Calling function *Called function*

Functions

Function prototype

A function prototype is simply the declaration of a function that specifies function's name, parameters and return type. It doesn't contain function body.

A function prototype gives information to the compiler that the function may later be used in the program.

Functions

Calling a function

Control of the program is transferred to the user-defined function by calling it.

Syntax of function call

```
functionName(argument1, argument2, ...);
```

In the above example, the function call is made using `addNumbers(n1, n2);` statement inside the `main()` function.

Functions

Passing arguments to a function

In programming, argument refers to the variable passed to the function. In the above example, two variables `n1` and `n2` are passed during the function call.

The parameters `a` and `b` accepts the passed arguments in the function definition. These arguments are called formal parameters of the function.

Functions

How to pass arguments to a function?

```
#include <stdio.h>

int addNumbers(int a, int b);

int main()
{
    ... .. ...

    sum = addNumbers(n1, n2);

    ... .. ...
}

int addNumbers(int a, int b)
{
    ... .. ...
    ... .. ...
}
```

If `n1` is of char type, `a` also should be of char type. If `n2` is of float type, variable `b` also should be of float type.

- The type of arguments passed to a function and the formal parameters must match, otherwise, the compiler will throw an error.
- A function can also be called without passing an argument.

Functions

Return Statement

The return statement terminates the execution of a function and returns a value to the calling function. The program control is transferred to the calling function after the return statement.

In the above example, the value of the `result` variable is returned to the main function. The `sum` variable in the `main()` function is assigned this value.

Functions

Syntax of return statement

```
return (expression);
```

For example,

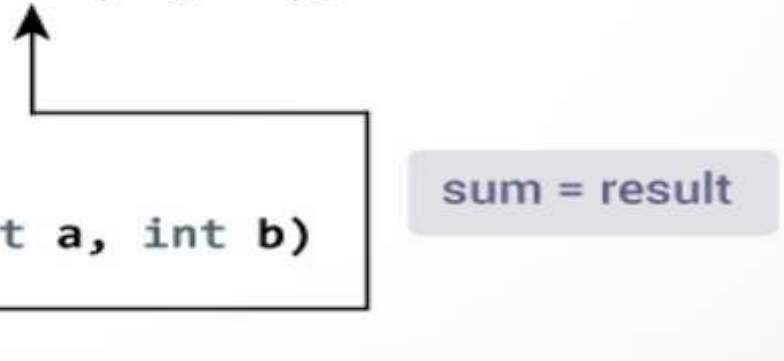
```
return a;
return (a+b);
```

The type of value returned from the function and the return type specified in the function prototype and function definition must match.

Functions

Return statement of a Function

```
#include <stdio.h>

int addNumbers(int a, int b);

int main()
{
    ... .. ...

    sum = addNumbers(n1, n2);

    ... .. ...
}

int addNumbers(int a, int b)
{
    ... .. ...
    return result;
}
```



sum = result


Thank You!