Introduction to functions

Comp Sci 1570 Introduction to C++

Outline

1 Basics
Return type
Function name
Parameter list
Function body
Return statement
Declaration, calling, and definition

2 Benefits

Functions

- A function is a reusable sequence of statements designed to do a particular job.
- In C++, a function is a group of statements that is given a name, and which can be called from some point of the program
- A function call is an expression that tells the CPU to interrupt the current function and execute another function.
- The CPU "puts a bookmark" at the current point of execution, and then calls (executes) the function named in the function call.
- When the called function terminates, the CPU goes back to the point it bookmarked, and resumes execution.
- The function initiating the function call is called the caller, and the function being called is the callee or called function.

Outline

1 Basics
Return type
Function name
Parameter list
Function body
Return statement
Declaration, calling, and definition

2 Benefits

Functions: return type


```
return_type function_name(parameter_list)
{
   local variable declarations;
   statements;
   return value_of_return_type;
}
```

- When you write your own functions, you get to decide whether a given function will return a value to the caller
- return_type is the data type of the value the function returns, which can be int, char, some pointer or even a class object.
- Some functions perform the desired operations without returning a value, with return_type is the keyword void.
- Return type does not indicate what specific value will be returned.
- It only indicates what type of value will be returned.

Outline

1 Basics
Return type
Function name
Parameter list
Function body
Return statement
Declaration, calling, and definition

2 Benefits

Functions: function name


```
return_type function_name(parameter_list)
{
   local variable declarations;
   statements;
   return value_of_return_type;
}
```

- Function Name : is the name of the function, using the function name it is called.

Outline

1 Basics
Return type
Function name
Parameter list
Function body
Return statement
Declaration, calling, and definition

2 Benefits

Functions: parameter list


```
return_type function_name(parameter_list)
{
   local variable declarations;
   statements;
   return value_of_return_type;
}
```

- Parameters are variables to hold values of arguments passed while function is called.
- Each parameter consists of a type followed by an identifier, with each parameter being separated from the next by a comma.
- Acts within the function as a regular variable which is local to the function.
- Allows passing arguments to the function from the location where it is called from.
- Parameters are optional and a function may contain none

Outline

1 Basics
Return type
Function name
Parameter list
Function body
Return statement
Declaration, calling, and definition

2 Benefits

Functions: function body


```
return_type function_name(parameter_list)
{
   local variable declarations;
   statements;
   return value_of_return_type;
}
```

- Block of statements surrounded by braces that specify what the function actually does

Outline

1 Basics
Return type
Function name
Parameter list
Function body
Return statement
Declaration, calling, and definition

2 Benefits

Functions: return statement


```
return_type function_name(parameter_list)
{
   local variable declarations;
   statements;
   return value_of_return_type;
}
```

- If a function has a non-void return type, it must return a value of that type (using a return statement).
- The only exception to this rule is for function main(), which will assume a return value of 0 if one is not explicitly provided.

Outline

Functions: declare, call, define


```
return_type function_name(parameter_list);

int main()
{
  function_name(parameter_list);
}

return_type function_name(parameter_list)
{
  local variable declarations;
  statements;
  return value_of_return_type;
}
```

- **Declare** above main
- **Call** in main
- **Define** after main

Outline

1 Basics
Return type
Function name
Parameter list
Function body
Return statement
Declaration, calling, and definition

2 Benefits

Why functions

- Break code down into smaller chunks
- Modularity
- Clarity
- Code-reuse
- Ease of update
- Hide function internals