Programming Language

Functions *(Day 2 Part 1)*

By: Bhaskar Ghosh
bjghosh@gmail.com

Updated: 4 August 2024

C++ Programming Language

Functions (Day 2 Part 1)

- Functions
- Library Functions and User Defined Functions
- Recursion
- Problem Solving using C++ [Level 2]
  - Logic Building and Debugging

C++ Programming Language

Functions

- A function is a block of code that performs a specific task.
  - Divide a complex problem into smaller chunks.
  - Make a program easy to understand and reusable.
- Types
  - Standard Library Functions: Predefined in C++
  - User-defined Function: Created by users
- Function Declaration Syntax:

```
returnType functionName (param1, param2,...);
```

- Function Definition Syntax:

```
returnType functionName (param1, param2,...)
{
    // function body
}
```

C++ Programming Language

User Defined Function

```
void hello () {
    cout << "Good Morning";
}
```

```
void main() {
    // calling a function
    hello();
}
```

- the name of the function is **hello**
- the return type of the function is **void**
- the parameter names along with the corresponding datatypes are enclosed within parentheses
- If a function doesn't have any parameters then it will have empty parenthesis
- the function body is written inside **{ }**

C++ Programming Language

Function Call

```
#include<iostream>

void displayNum(int n1, double n2) {
    // code
}

int main() {
    ... ...
    displayNum(num1, num2);
    ... ...
}
```

function call

- **num1** and **num2** are passed as arguments.
- These values are stored by the function parameters **n1** and **n2** respectively.

C++ Programming Language

Return Statement

```
#include<iostream>

int add(int a, int b) {
    return (a + b);
}

int main() {
    int sum;

    sum = add(100, 78);
    ... ...
}
```

function call

- Returned value of the function is stored in the variable **sum**
- So, datatype of variable **sum** should be able to store the return type of the function.

C++ Programming Language

Library Functions

- Library functions are the built-in functions in C++ programming.
- Some common library functions in C++ are sqrt(), abs(), isdigit(), etc.
- For example, explore the library functions in the **`cmath`** header file.
  - https://www.programiz.com/cpp-programming/library-function/cmath
  - https://www.programiz.com/cpp-programming/library-function

C++ Programming Language

Default Arguments

- In C++ programming, we can provide default values for function parameters.
- If a function with default arguments is called without passing arguments, then the default parameters are used.
- But, if arguments are passed while calling the function, the default arguments are ignored.

C++ Programming Language Default Arguments

Case 1 : No argument is passed

```
void temp(int = 10, float = 8.8);

int main() {
  ... ...
  temp();
  ... ...
}

void temp(int i, float f) {
  // code
}
```

Case 2 : First argument is passed

```
void temp(int = 10, float = 8.8);

int main() {
  ... ...
  temp(6);
  ... ...
}

void temp(int i, float f) {
  // code
}
```

Case 3 : All arguments are passed

```
void temp(int = 10, float = 8.8);

int main() {
  ... ...
  temp(6, -2.3);
  ... ...
}

void temp(int i, float f) {
  // code
}
```

Case 4 : Second argument is passed

```
void temp(int = 10, float = 8.8);

int main() {
  ... ...
  temp(3.4);
  ... ...
}

void temp(int i, float f) {
  // code
}
```

Error

C++ Programming Language

Default Arguments

- Once we provide a default value for a parameter, all subsequent parameters must also have default values.
  - `void add(int a, int b = 3, int c, int d); // Invalid`
  - `void add(int a, int b = 3, int c, int d = 4); // Invalid`
  - `void add(int a, int c, int b = 3, int d = 4); // Valid`
- If we are defining the default arguments in the function definition instead of the function prototype, then the function must be defined before the function call.
  -
```
// Invalid code
int main() {
    display(); // function call
}
void display(char c = '*', int count = 5) {
    // code
}
```

C++ Programming Language

Inline Function

- Declaring a function as inline copies the function code to the location of the function call in compile-time.
- To create an inline function, we use the inline keyword.

```
inline returnType functionName(parameters) {
    // code
```

```
inline void displayNum(int num) {
  cout << num << endl;
}

int main() {

  displayNum(5);

  displayNum(8);

  displayNum(666);
}
```

Compilation →

```
inline void displayNum(int num) {
  cout << num << endl;
}

int main() {

  cout << 5 << endl;

  cout << 8 << endl;

  cout << 666 << endl;
}
```

C++ Programming Language

Recursion

- A function that calls itself is known as a recursive function.
  - And, this technique is known as recursion.
- Using recursion, it is possible to solve a complex problem with very few lines of code
  - By dividing the input of the problem statement with each function call
  - Until finally stopping to return/provide the combined solution.
- Example:
  - Find the factorial of a given number.

Programming Language

Arrays and Strings *(Day 2 Part 2)*

By: Bhaskar Ghosh
bjghosh@gmail.com

Updated: 4 August 2024

C++ Programming Language

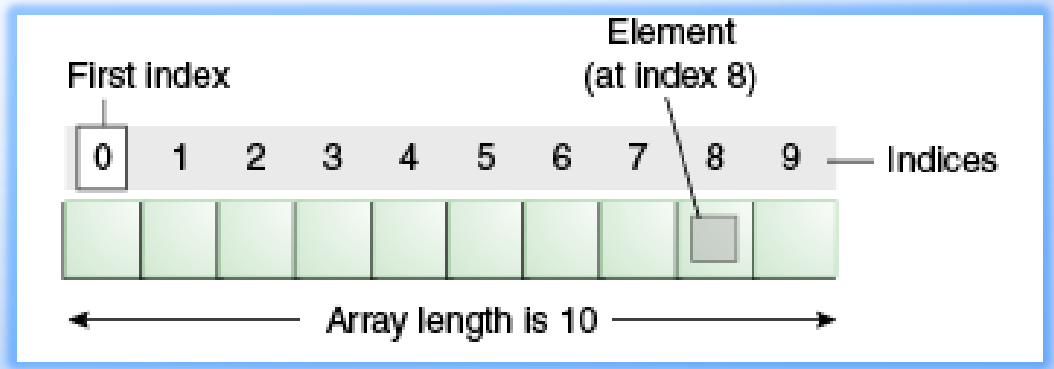
Arrays and Strings (Day 2 Part 2)

- C++ Arrays
  - Single Dimensional, and Multi Dimensional
  - Length of an array
  - Passing array to a function – array decay
- C++ Strings
- C++ string class
- Problem Solving using C++ [Level 2]
  - Logic Building and Debugging

C++ Programming Language

Arrays

- Arrays store data of same data type.
- The elements of an array are stored in a contiguous memory location.
- Array size remains constant once declared.
- Index-based
  - 1st element of the array is stored at the 0th index.
  - 2nd element is stored on 1st index and so on.
- Syntax
  - `dataType arrayName[arraySize];`
- Example
  - `int marks[10];`



C++ Programming Language

Initializing Arrays in C++

Initialization Type	Example	Description
Initialize Array with Values in C++	int arr[5] = {1, 2, 3, 4, 5};	Initializes an array with specified values.
Initialize Array with Values and without Size	int arr[] = {1, 2, 3, 4, 5};	Array size is determined by the number of elements.
Initialize Array after Declaration (Using Loops)	for (int i = 0; i < N; i++) { arr[i] = value; }	Initializes array elements using a loop, useful for user input.
Initialize an array partially in C++	int arr[5] = {1, 2};	Initializes only the specified elements, others are set to 0.
Initialize the array with zero in C++	int arr[5] = {0};	Initializes all elements to 0, can be overridden individually.

C++ Programming Language

Arrays

- Array Declaration

```
dataType arrayName[arraySize];
```

```
int x[6];
```

- Array Initialization

```
dataType arrayName[arraySize] = {comma separated values};
```

```
int x[6] = {19, 10, 8, 17, 9, 15};
```

- Accessing Array Elements
  - `arrayName[index];`

C++ Programming Language

Multidimentional Arrays

- In C++, we can create an array of an array, known as a multidimensional array.
  - `dataType arrayName[size1d][size2d]...[sizeNd];`
- `int x[3][4];`
  - `x` is a two-dimensional array.
  - It can hold a maximum of 12 elements.
  - We can access an element as
    `x[i][j];`
    where `i` *is the row number,* `j` *is the column number*

	Col 1	Col 2	Col 3	Col 4
Row 1	x[0][0]	x[0][1]	x[0][2]	x[0][3]
Row 2	x[1][0]	x[1][1]	x[1][2]	x[1][3]
Row 3	x[2][0]	x[2][1]	x[2][2]	x[2][3]

- `int y[3][4][4];`
  - `y` is a three-dimensional array.

C++ Programming Language

Arrays – size of an array, passing array to function, and array decay

- We can use the sizeof operator to find the size of an array.
  - How do we find out how many elements are stored in an array?

```
sizeof(x)/sizeof(x[i]);
```

- In C++, array decays into a pointer after being passed as a parameter to a function.
  - How do we pass an array to a function?
    - `void func(int x[5]);`
    - `void func(int x[]);`
    - `void func(int* x);`
  - What is meant by array decay?
    - Passing an array to function results in array decay due to which the array loses information about its size.
    - i.e., size of the array or the number of elements of the array cannot be determined anymore.

C++ Programming Language

Strings in C++

C style string

```
char s[] = "abc";
char s[] = {'a', 'b', 'c'};
char* s = "abc";
```

C++ style string

```
string s("abc");
string s = "abc"
```

- A string in C++ is an object that represents a sequence of characters.
- But, a string in C is represented by an array of characters, ending with a null character `'\0'`.
- C++ also supports C style strings.
- Strings in C++ are a part of the standard string class (**`std::string`**).

C++ Programming Language

Strings in C++

C++ string	Char Array (C style)
A string is a **class that defines objects** that can be represented as a stream of characters.	A character array is simply an **array of characters** that can be terminated by a null character.
In the case of strings, memory is **allocated dynamically**. More memory can be allocated at run time on demand. As no memory is pre-allocated, **no memory is wasted**.	The size of the character array has to be **allocated statically,** more memory cannot be allocated at run time if required. Unused allocated **memory is also wasted**
As strings are represented as objects, **no array decay** occurs.	There is a **risk of array decay** in the case of the character array.
Strings are slower when compared to implementation than character array.	Implementation of **character array is faster** than std:: string.
String class defines **many inbuilt functions** that allow various operations on strings.	Character arrays **do not offer** many inbuilt functions to manipulate strings.

C++ Programming Language

Strings in C++

- How to read a string entered by the user (standard input e.g. keyboard)?
  - cin >> str;
  - cin.get(str, 50);
  - cin.getline(str, 50);
- Lookup the string class methods available in C++
  - https://www.w3schools.com/cpp/cpp_ref_string.asp
  - https://cplusplus.com/reference/string/string/

C++ Programming Language

C++ String Operations

- Capacity Functions
  - length()
  - capacity()
  - resize()
  - shrink_to_fit()
- Iterator Functions
  - begin()
  - end()
  - rbegin()
  - rend()
- Manipulating Functions
  - copy()
  - swap()
  - append()
  - find()
  - rfind()
  - replace()
  - substr()
- Element Access
  - at()

C++ Programming Language

C++ String Iterator Functions

Function	Definition
begin()	This function returns an iterator to the beginning of the string.
end()	This function returns an iterator to the next to the end of the string.
rbegin()	This function returns a reverse iterator pointing at the end of the string.
rend()	This function returns a reverse iterator pointing to the previous of beginning of the string.
cbegin()	This function returns a constant iterator pointing to the beginning of the string, it cannot be used to modify the contents it points-to.
cend()	This function returns a constant iterator pointing to the next of end of the string, it cannot be used to modify the contents it points-to.
crbegin()	This function returns a constant reverse iterator pointing to the end of the string, it cannot be used to modify the contents it points-to.
crend()	This function returns a constant reverse iterator pointing to the previous of beginning of the string, it cannot be used to modify the contents it points-to.

C++ Programming Language

Problem Solving using C++ [Level 2]

logic Building and Debugging

C++ Programming Language

Problem Solving using C++ [Level 2] – (Functions, Recursion, Arrays)

1. Modify the Calculator program logic into a function, and call the function from main() method based on user choice.+
2. WAF to print the largest of 3 numbers, and call the function passing 3 user inputted values.
3. WAF to find GCD using recursion.
4. WAP to find sum of elements in a given array.
5. WAP to find the largest element in an array.
6. WAF to reverse an array.
7. WAF to find a given value from an array.
8. WAP to Segregate 0s and 1s in an array.

WAP to print Fibonacci Series using recursion.

C++ Programming Language

Problem Solving using C++ [Level 2] – (Strings)

1. WAP to swap two strings.
2. WAP to check if a given string is Palindrome.
3. WAP to reverse an array.
4. WAP to print reverse of a string using recursion.
5. WAP to count words in a given string.
6. WAP to reverse words in a given string.
7. WAP to remove “b” and “ac” from a given string.
8. WAP to write your own atoi() function.
   - The atoi() function in C takes a string (which represents an integer) as an argument and returns its value as type int.