Basics of Data Structure

By Dr. Sanjeev Kumar Mandal

Aim

To equip the students with the basic skills of using Data Structures in programs

Instructional Objectives

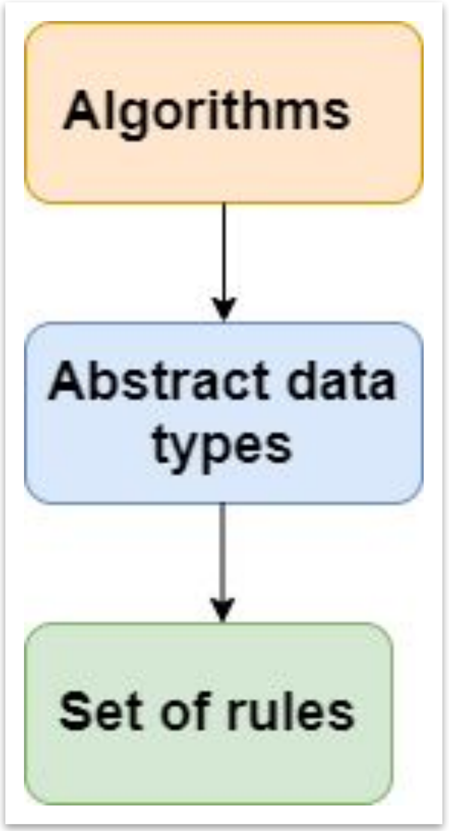
Objectives of this chapter are:

- Describe Data Structures and its types
- Explain items included elementary data organisation
- Summarize the role of algorithms in programming
- Explain the procedure to calculate time and space complexities
- Explain the string processing with its functions
- Demonstrate memory allocation and address variable

Data Structures and its Types

Definition

- The data structure name indicates **organizing the data in memory**.
- There are **many ways of organizing the data in the memory** as we have **already seen one of the data structures, i.e., array in C language**.
- **Array is a collection of memory elements in which data is stored sequentially**, i.e., one after another.
- In other words, we can say that **array stores the elements in a continuous manner.**



GOALS OF DATA STRUCTURES

Data structure **stores and organizes a large set of data in computer memory**. It **ensures the reusability of data efficiently later**. Data can be **stored logically or mathematically**. It can vary depending on the organizational needs and adaptability levels.

The data models are adapted based on two factors:

1. **The structure was established to reflect the relation of data with real-world objects.**
2. **Low level of complexity for reusability and robustness.**

Desired Characteristics of a Data Structure:

- **Correctness:** By correctness, we mean that a **data structure is designed, to work correctly** for all possible inputs that one might encounter.
- **Efficiency**: Useful **data structure** and their operations also need to be **efficient** i.e., they should be **fast** and not use more of the **computer's resources**, such **as memory space**, than required.
- **Robustness**: Every good programmer wants to produce software that is robust, which means **that a program produces the correct output for all inputs**

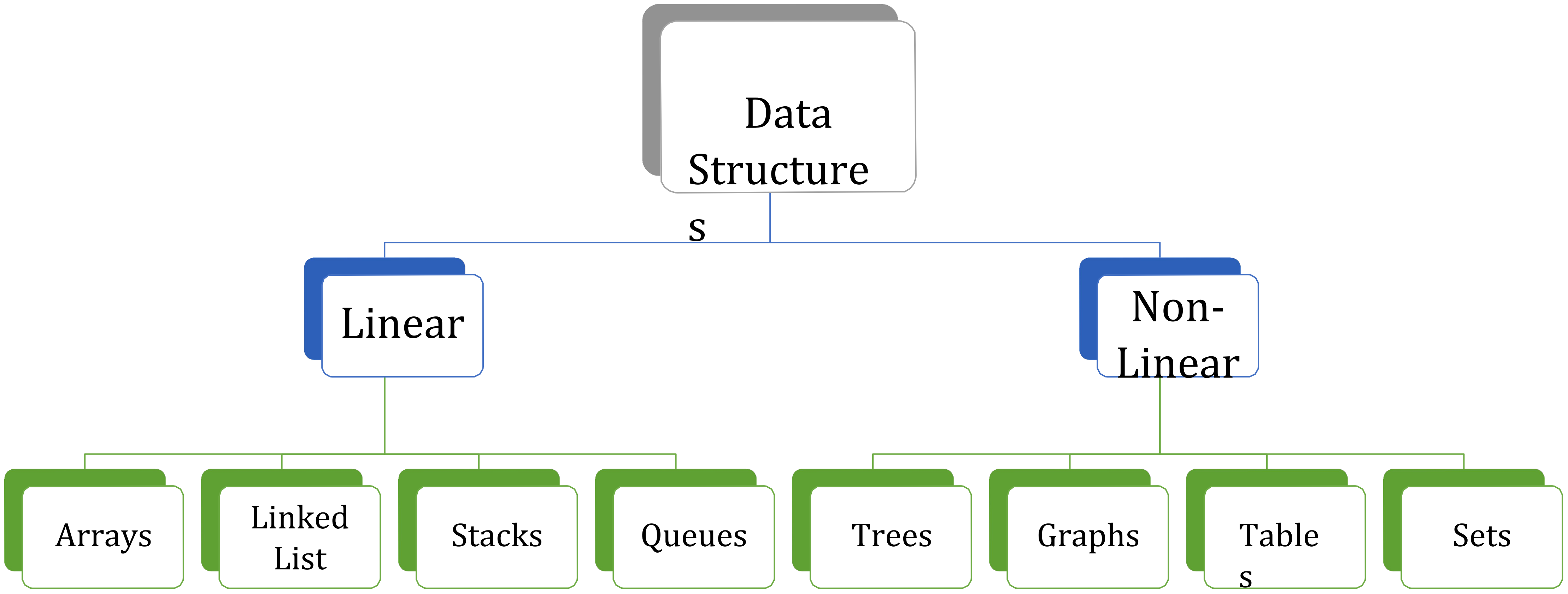
Adaptability: Software, therefore, needs to be able to evolve over time in response to changing conditions. These change scan be expected, such as the need to adapt to an increase in CPU speed. Software should also be able to adapt to unexpected events. Thus, another important goal of quality software is that it be adaptable.

Reusability: Developing quality software can be expensive, and its cost can be reduced somewhat if the software is designed in a way that make sit easily reusable in future applications. Software reuse can be a significant cost-saving and timesaving technique.

Popular data structures and their usage

- **Arrays:** Array stores elements with the same data types in contiguous memory locations.
- **Linked list:** Linked list contains nodes storing data and memory attached to it. The data is stored in a contiguous memory location and hence the address stored in the node links it to the next address where the data is stored
- **Stack**: Stack is a linear data structure that allows insertion and deletion through the open end. The insertion operation is known as **push** while deletion is called **pop**
- **Queue:** Another linear data structure where the operation follows the **first in first out rule**. Insertion in queues is known as **enqueue** and deletion is known as **dequeue**
- **Graph:** A graph is a non-linear data structure with **nodes and edges**. It **connects other nodes in the graph and forms a graphical image of an object**. The graph can be directed or undirected. Directed graphs are connected in a single direction while undirected graphs are bidirectional
- **Trees:** Trees are non-linear data structures. It forms **hierarchical relations** with the data elements. The root node is the point from which the tree descends. A tree might have subtrees. It exhibits the parent-child relationship. The node that descends from a certain node is called a child node. A parent node can have multiple child nodes but only one parent node

Types of Data Structures



	Linear Data structure	Non-Linear Data structure
Basic	In this structure, the elements are arranged sequentially or linearly and attached to one another.	In this structure, the elements are arranged hierarchically or non-linear manner.
Types	Arrays, linked list, stack, queue are the types of a linear data structure.	Trees and graphs are the types of a non-linear data structure.
implementation	Due to the linear organization, they are easy to implement.	Due to the non-linear organization, they are difficult to implement.
Traversal	As linear data structure is a single level, so it requires a single run to traverse each data item.	The data items in a non-linear data structure cannot be accessed in a single run. It requires multiple runs to be traversed.
Arrangement	Each data item is attached to the previous and next items.	Each item is attached to many other items.
Levels	This data structure does not contain any hierarchy, and all the data elements are organized in a single level.	In this, the data elements are arranged in multiple levels.
Memory utilization	In this, the memory utilization is not efficient.	In this, memory is utilized in a very efficient manner.
Time complexity	The time complexity of linear data structure increases with the increase in the input size.	The time complexity of non-linear data structure often remains same with the increase in the input size.
Applications	Linear data structures are mainly used for developing the software.	Non-linear data structures are used in **image processing** and **Artificial Intelligence**.

Types of Data Structures

1. **Primitive Data Structures:**

   They are directly operated upon by machine level instructions.

   Examples: int, float double etc.

2. **Non-primitive data structure:**

   These data types are derived from primary data types

   They are used to store group of values.

   Examples: Arrays, Structures, Linked Lists, Queues etc.

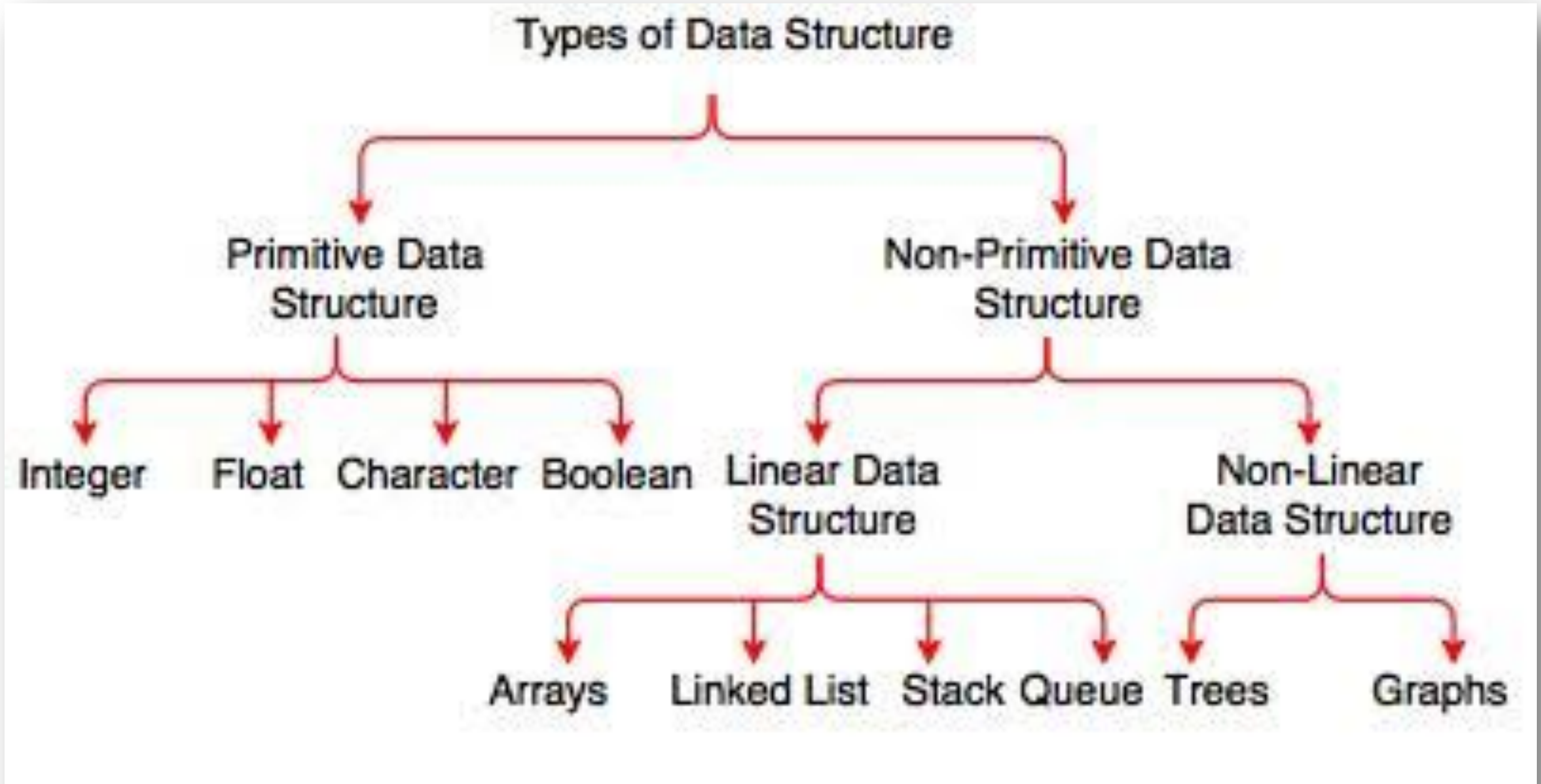


Fig. Types of Data Structure

Quiz / Assessment

4) Which of the following data structure is linear type?

a) Graph
b) Trees
c) Binary tree
d) Stack

5) Which of the following data structure is non-linear type?

a) Strings
b) Lists
c) Stacks
d) Graph

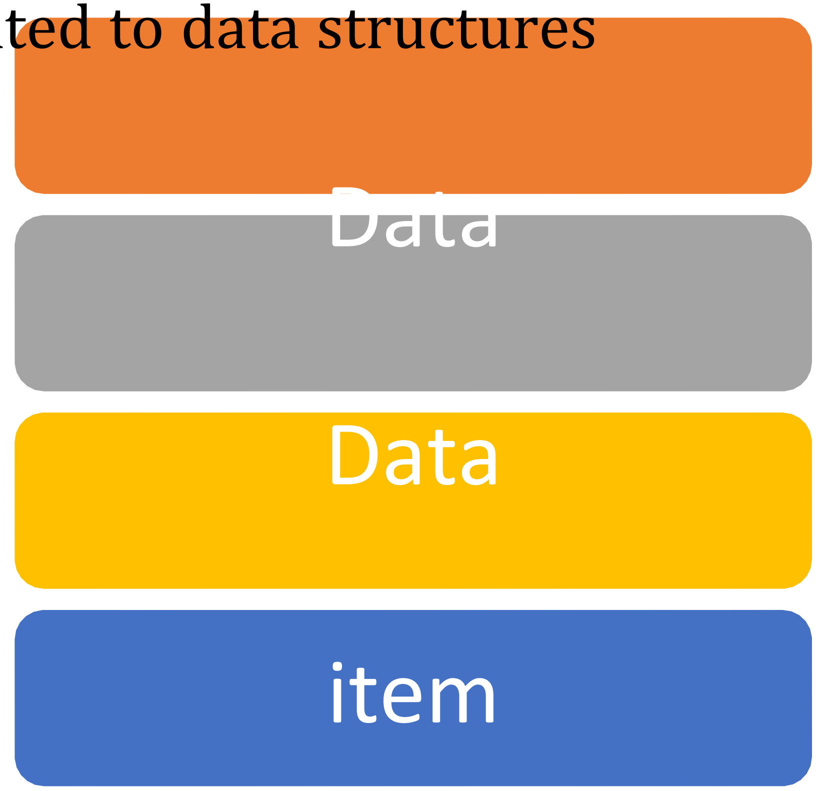
6) Which of the following data structure can't store the non-homogeneous data elements?

a) Arrays
b) Records
c) Pointers
d) Stacks

Items Included in Elementary Data Organisation

Items in Elementary Data Organization

Basic Terminologies related to data structures are:



Entity SET

Table Name : Student

Entity : Each row is an entity.

Example : **1 Avi 19 M**

Student_ID	Student_Name	Student_Age	Student_Gender
1	Avi	19	M
2	Ayush	23	M
3	Nikhil	21	M
4	Riya	16	F

Items in Elementary Data Organization (Contd.)

Informatio

n Field

Record

File

Ke

Role of Algorithms in Programming

Algorithms – Role in Programming

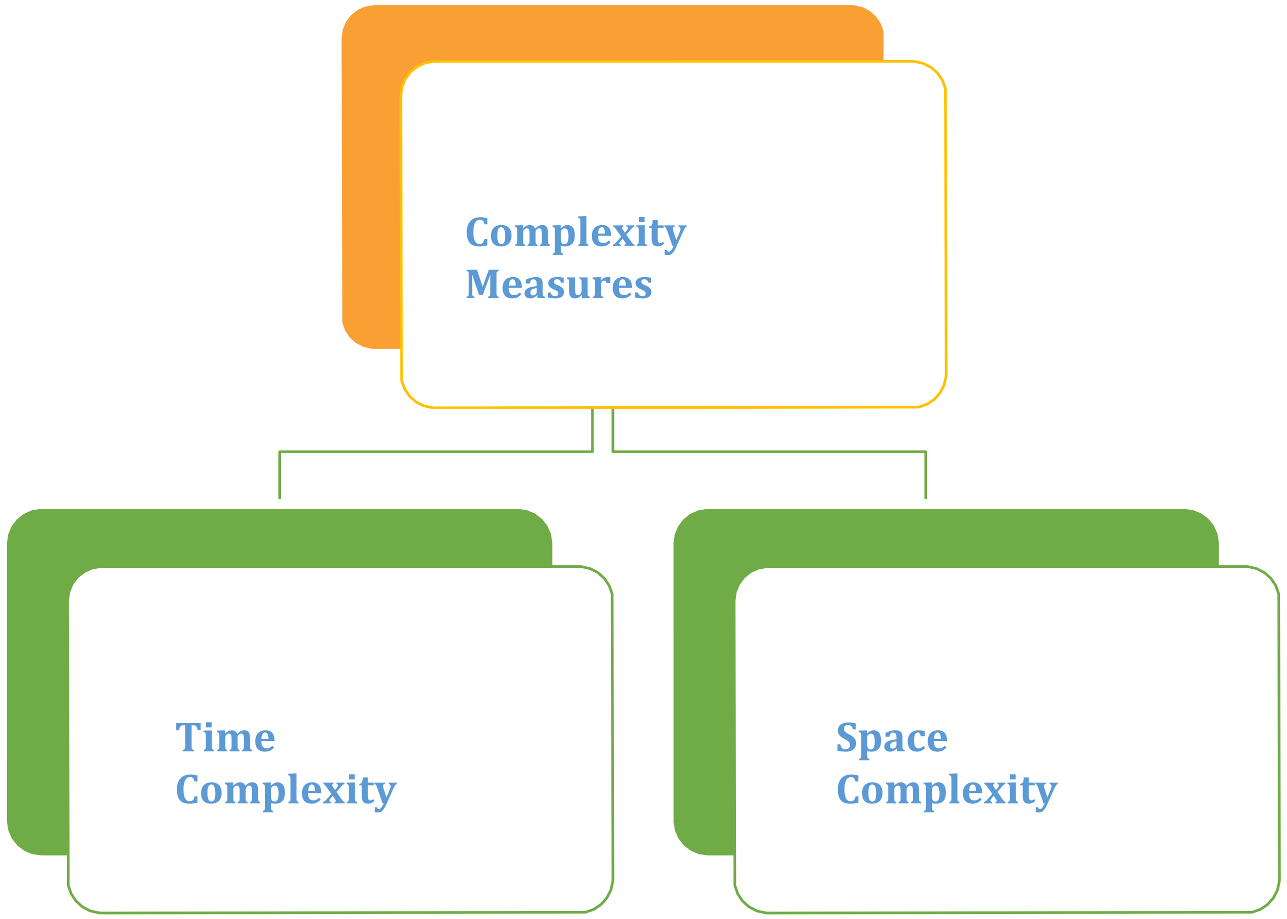
An algorithm is a step by step process to solve a problem.

In programming, algorithms are implemented in the form of methods

or functions or routines.

Procedure to Calculate Time and Space Complexities

Time and Space Complexity



Time Complexity

It describes amount of time an algorithm takes in terms of amount of input to the algorithm.

Expressed in :

- Number of memory access made
- No. of comparisons between two integers
- Number of times a particular loop got executed

Time complexity is not "wall clock" time.

Space Complexity

- It is a function which describes the amount of memory utilized by an algorithm in terms of the

  amount of input to the algorithm.

- We use natural, but fixed-length units, to measure space complexity.

String Processing with its Functions

String Processing

Textual data is represented using arrays of characters called a *string*.

End of string is marked with a null character.

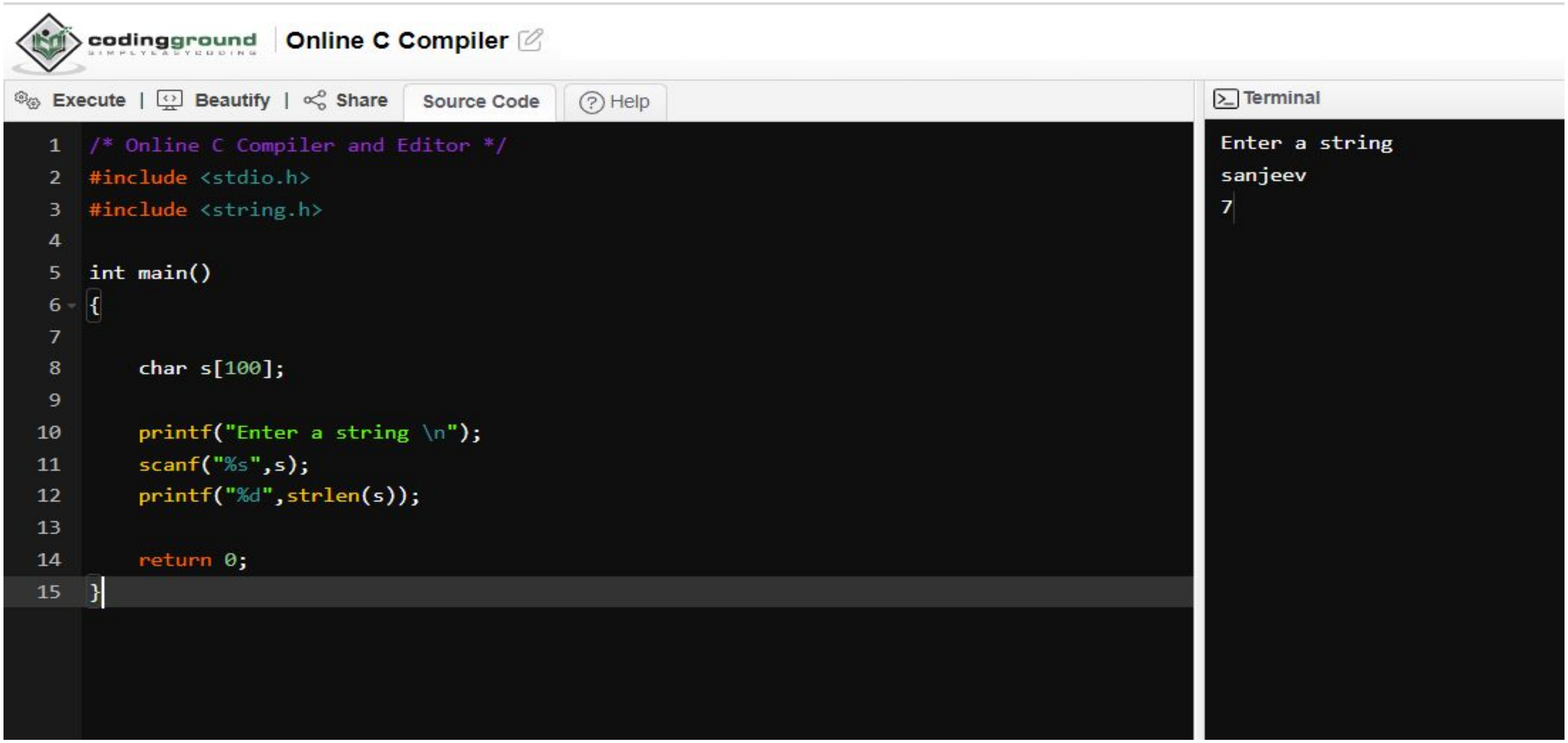
The null or string-terminating character is represented by another character escape sequence, \0.

Eg: char string[] = "Hello, world!";

Commonly used string handling functions are:

- Strlen()
- Strcpy()
- Strcat()
- Strcmp()
- Strlwr()
- Strupr()

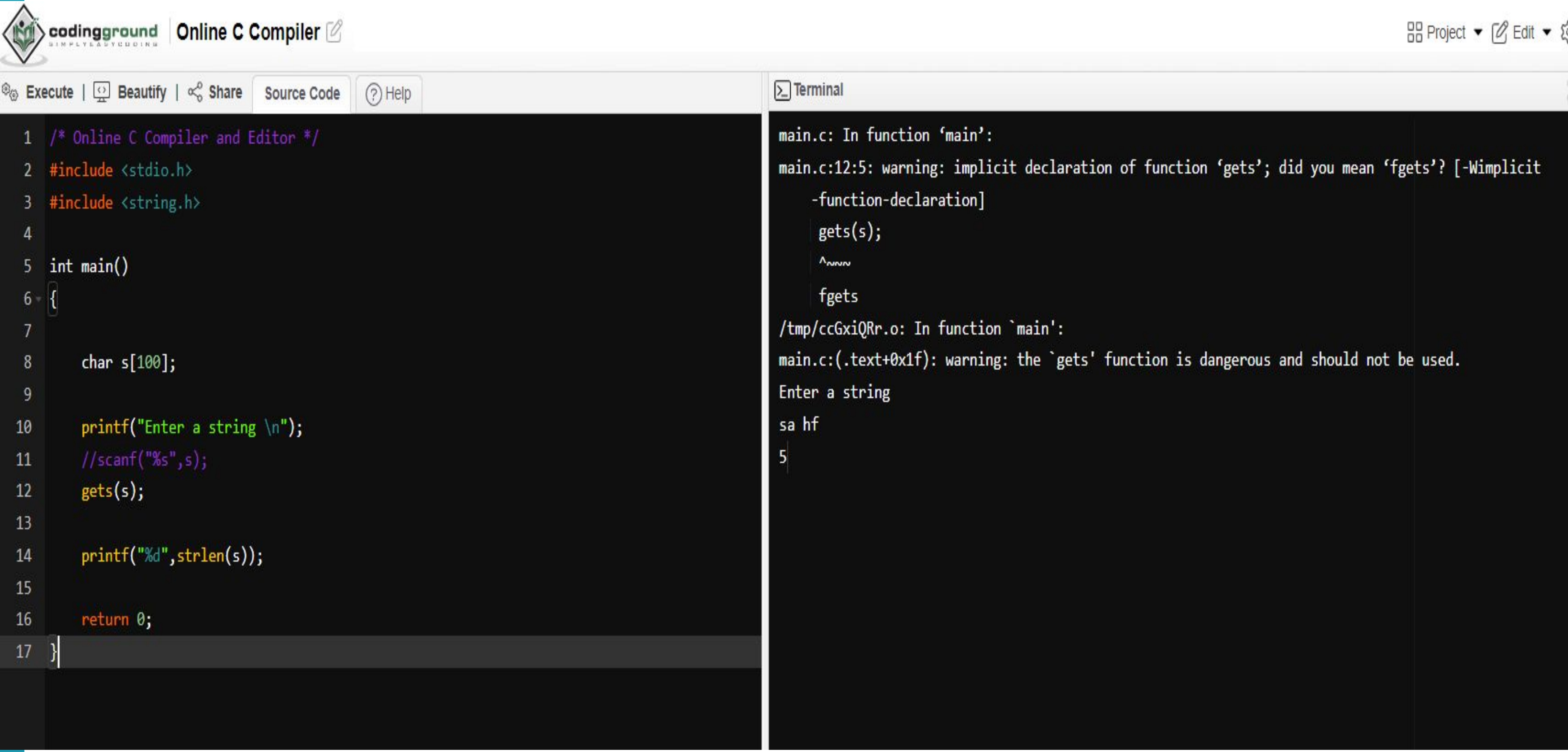
Without Space



```c
/* Online C Compiler and Editor */
#include <stdio.h>
#include <string.h>

int main()
{

    char s[100];

    printf("Enter a string \n");
    scanf("%s",s);
    printf("%d",strlen(s));

    return 0;
}
```

```
Enter a string
sanjeev
7
```

With Space



```c
/* Online C Compiler and Editor */
#include <stdio.h>
#include <string.h>

int main()
{

    char s[100];

    printf("Enter a string \n");
    //scanf("%s",s);
    gets(s);

    printf("%d",strlen(s));

    return 0;
}
```

```
main.c: In function ‘main’:
main.c:12:5: warning: implicit declaration of function ‘gets’; did you mean ‘fgets’? [-Wimplicit
    -function-declaration]
     gets(s);
     ^~~~
     fgets
/tmp/ccGxiQRr.o: In function `main':
main.c:(.text+0x1f): warning: the `gets' function is dangerous and should not be used.
Enter a string
sa hf
5
```

More string processing

Function	Description
Strcmpi()	Compares two strings with case insensitivity
Strrev()	Reverses a string
Strlwr ()	Converts an string letters to lowercase
Strupr()	Converts lowercase string letters to uppercase
Strchr()	Finds the first occurrence of a given character in a string
Strrchr()	Finds the last occurrence of a given character in a string
Strset()	Sets all characters in a string to a given character
Strnset()	Sets the specified number of characters ina string to a given character
Strdup()	Used for duplicating a string

strrrv()

- The strrev() function is a built-in function in C and is defined in string.h header file. The strrev() function is used to reverse the given string.

- **Syntax:**

- char *strrev(char *str);

- **Example:**
- #include <stdio.h>
- #include <string.h>
- int main()
- {
-     char str[50] = "123456789";
-
-     printf("The given string is =%s\n", str);
-
-     printf("After reversing string is =%s", strrev(str));
-      return 0;
- }

strlwr()

- The strlwr( ) function is a built-in function in C and is used to convert a given string into lowercase.
- Syntax:
- char *strlwr(char *str);

- **Example**
- #include<stdio.h>
- #include <string.h>
-
- int main()
- {
-   char str[] = "CompuTer ScienCe PoRTAl fOr geeKS";
-     printf("Given111 string is: %s\n",str);
-
-   printf("\nString after converting to the "
-            "lowercase is: %s",strlwr(str));
-     return 0;
- }

Strupr()

- The strupr( ) function is used to converts a given string to uppercase.
- Syntax:
- char *strupr(char *str);

Example:

```
#include<stdio.h>
#include <string.h>

int main()
{
  char str[] = "CompuTer ScienCe PoRTAl fOr
geeKS";

  printf("Given string is: %s\n", str);

  printf("\nstring after converting to the
uppercase is: %s", strupr(str));
  return 0;
}
```

Strchr()

- The C library function char *strchr(const char *str, int c) searches for the first occurrence of the character c (an unsigned char) in the string pointed to by the argument str.

- Declaration
- char *strchr(const char *str, int c)

```
• #include <stdio.h>
• #include <string.h>
• int main ()
• {
•    const char str[] = "Use strchr() function in C.";
•    const char ch = 's'; // it is searched in str[]
•    char *ptr; // declare character pointer ptr
•          printf (" Original string is: %s \n", str);
•    // use strchr() function and pass str in which
  ch is to be searched
•    ptr = strchr( str, ch);
•    printf (" The first occurrence of the '%c' in '%s'
  string  is: '%s' ", ch, str, ptr);
•    return 0;
• }
```

Strrchr()

- The C library function char *strrchr(const char *str, int c) searches for the last occurrence of the character c (an unsigned char) in the string pointed to, by the argument str.

- Declaration
- Following is the declaration for strrchr() function.
- char *strrchr(const char *str, int c)

- **Example:**
- #include <stdio.h>
- #include <string.h>
- int main ()
- {
-   int len;
-   const char str[] = "http://www.tutorialspoint.com";
-   const char ch = '.';
-   char *ret;
-   ret = strrchr(str, ch);
-   printf("String after |%c| is - |%s|\n", ch, ret);
-   return(0);
- }

Strnset()

- The strnset() function is a built-in function in C and it sets the first n characters of a string to a given character. If n is greater than the length of string, the length of string is used in place of n.
- Syntax:
- char *strnset(const char *str, char ch, int n);

- **Example:**
- #include <stdio.h>
- #include <string.h>
-   int main()
- {
-     char str[] = "GeeksforGeeks";
-
-     printf("Original String: %s\n", str);
-
-     // First 5 character of string str
-     // replaced by character '*'
-     printf("Modified String: %s\n", strnset(str, '*', 5));
-         return 0;
- }

Strnset()

- The strnset() function is a builtin function in C and it sets the first n characters of a string to a given character. If n is greater than the length of string, the length of string is used in place of n.
- Syntax:
- char *strnset(const char *str, char ch, int n);

- **Example:**
- #include <stdio.h>
- #include <string.h>
-   int main()
- {
-     char str[] = "Computer Science";
-     printf("Original String: %s\n", str);
-         // First 5 character of string str
-     // replaced by character '*'
-     printf("Modified String: %s\n", strnset(str, '*', 5));
-
-     return 0;
- }

- The **strdup()** and functions is used to duplicate a string.
  strdup() :
- Syntax : *char *strdup(const char *s);*
- This function returns a pointer to a null-terminated byte string, which is a duplicate of the string pointed to by ***s***. The memory obtained is done dynamically using malloc and hence it can be freed using free().

- **Example:**
- #include<stdio.h>
- #include<string.h>
- int main()
- {
-     char source[] = "GeeksForGeeks";
-      // 5 bytes of source are copied to a new memory
-     // allocated dynamically and pointer to copied
-     // memory is returned.
-     char* target = strndup(source, 5);
-      printf("%s", target);
-     return 0;
- }

Quiz / Assessment

10)Which of the following function compares 2 strings with case-insensitively?

a)Strcmp(s, t)

b)strcmpcase(s, t)

c) Strcasecmp(s, t)

d) strchr(s, t)

11)How will you print \n on the screen?

a)printf("\n")

b) printf('\n');";

c)echo "\\n

d)printf("\\n");

12) Strcat function adds null character

a) Only if there is space

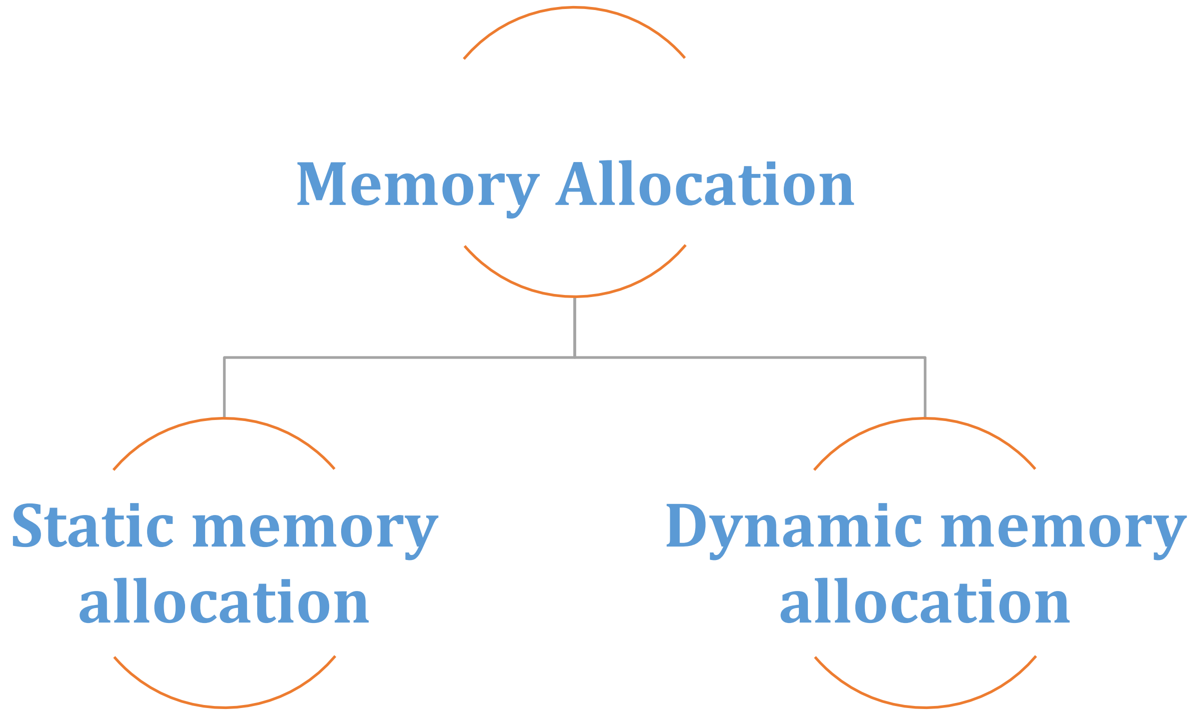
b) Always

c)depends on the standard

d)Depends on the compiler

Memory Allocation and Address Variable

Memory Allocation



Static Memory Allocation

- Compiler allocates required memory space
- Every variable declared in program are allocated unique memory addresses
- The variables may be assigned with a pointer variable
- Memory should be reserved in advance.
- Not possible to allocate memory during Runtime

Dynamic Memory Allocation

- Memory need not be Reserved in advance
- Executing program request processor for a memory block during its run time
- It enables us to create data types and structures of any size and length to suit our programs need within the program.
- Doesn't waste memory space.

Quiz / Assessment

13) In static memory allocation, complier allocates required memory using dereference operator

a) True b) False

14) Memory is dynamically allocated once the program is complied.

b) True b) False

15) Memory Fragmentation occurs when small holes of memory are formed which cannot be used to fulfil the dynamic requests.

b) True b) False

Quiz / Assessment

16) Choose the right answer. Prior to using a pointer variable

a) It should be declared b) It should be initialized

c) It should be declared and initialized d) It should be neither declared nor initialized

17) The address operator &, cannot act on and

18) The operator > and < are meaningful when used with pointers, if

a) The pointers point to data of similar type

b) The pointers point to structure of similar data type.

c) The pointers point to elements of the same array

d) The pointers point to elements of the another array

Activity

Brief description of

Offline Activity
(15 min)

- Divide the students into two groups.
- Give selection sort algorithm and insertion sort algorithm to each of the groups.
- Students should calculate the time complexities for both these algorithms.

Summary

- ✔ Data Structure is a way of collecting and organising data in such a way that we can perform operations on these data in an effective way.
- ✔ Data structures are categorized into two types: linear and nonlinear. Linear data structures are the ones in which elements
  are arranged in a sequence, nonlinear data structures are the ones in which elements are not arranged sequentially
- ✔ The complexity of an algorithm is a function describing the efficiency of the algorithm in terms of the amount of data the algorithm must process.
- ✔ The basic terminologies in the concept of data structures are Data, Data item, Entity, Entity set, Information, Record, file, key etc.
- ✔ String functions like strcmp, strcat, strlen are used for string processing in C
- ✔ Static and Dynamic memory allocation are the core types of memory allocation
- ✔ The address of a variable can be obtained by preceding the name of a variable with the address-of operator (&)

e-References

- cs.utexas.edu, (2016). Complexity Analysis. Retrieved on 19 April 2016, from

  https://www.cs.utexas.edu/users/djimenez/utsa/cs1723/lecture2.html

- compsci.hunter.cuny.edu, (2016). C Strings and Pointers. Retrieved on 19 April 2016, from

  http://www.compsci.hunter.cuny.edu/~sweiss/resources/cstrings.pdf

- Msdn.microsoft.com, (2016). An extensive examination of Data Structures. Retrieved on 19 April 2016, from
  https://msdn.microsoft.com/en-us/library/aa289148(v=vs.71).aspx

External Resources

1. Kruse, R. (2006). *Data Structures and program designing using 'C'* (2nd ed.). Pearson Education.
2. Srivastava, S. K., & Srivastava, D. (2004). *Data Structures Through C in Depth* (2nd ed.). BPB Publications.
3. Weiss, M. A. (2001). *Data Structures and Algorithm Analysis in C* (2nd ed.). Pearson Education.

Thank you