**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **23-June-2020** | **Name:** | **Raziya Banu** |
| **Course:** | **C++ Programming** | **USN:** | **4AL16EC058** |
| **Topic:** | **Data types, Array and pointers** | **Semester & Section:** | **8th sem & ‘B’ section** |
| **Github Repository:** |  |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
| **Image of session** |
| **Report –**  In my first session today I have studied about - Data types, Array and pointers    [Pointer to an array](https://www.geeksforgeeks.org/pointer-array-array-pointer/) is also known as [array pointer](https://www.geeksforgeeks.org/pointer-array-array-pointer/). We are using the pointer to access the components of the array.  int a[3] = {3, 4, 5 };  int \*ptr = a;  We have a pointer ptr that focuses to the 0th component of the array. We can likewise declare a pointer that can point to whole array rather than just a single component of the array. **Syntax**:  data type (\*var name)[size of array];  **Declaration of the pointer to an array:**  // pointer to an array of five numbers  int (\* ptr)[5] = NULL;  The above declaration is the pointer to an array of five integers. We use parenthesis to pronounce pointer to an array. Since subscript has higher priority than indirection, it is crucial to encase the indirection operator and pointer name inside brackets.  **Example:**  filter\_none  edit  play\_arrow  brightness\_4   |  | | --- | | // C program to demonstrate  // pointer to an array.  #include <stdio.h>    int main()  {        // Pointer to an array of five numbers      int(\*a)[5];        int b[5] = { 1, 2, 3, 4, 5 };        int i = 0;        // Points to the whole array b      a = &b;        for (i = 0; i < 5; i++)            printf("%d\n", \*(\*a + i));        return 0;  } |   **Output:**  1  2  3  4  5  [**Array of pointers**](https://www.geeksforgeeks.org/pointers-in-c-and-c-set-1-introduction-arithmetic-and-array/):  “[Array of pointers](https://www.geeksforgeeks.org/pointers-in-c-and-c-set-1-introduction-arithmetic-and-array/)” is an array of the [pointer variables](https://www.geeksforgeeks.org/pointers-c-examples/). It is also known as pointer arrays. **Syntax**:  int \*var\_name[array\_size];  Declaration of an array of pointers:  int \*ptr[3];  We can make separate pointer variables which can point to the different values or we can make one integer array of pointers that can point to all the values.  **Example:**  filter\_none  edit  play\_arrow  brightness\_4   |  | | --- | | // C program to demonstrate  // example of array of pointers.    #include <stdio.h>    const int SIZE = 3;    void main()  {      int arr[] = { 1, 2, 3 };        // we can make an integer pointer array to      // storing the address of array elements      int i, \*ptr[SIZE];        for (i = 0; i < SIZE; i++) {            // assigning the address of integer.          ptr[i] = &arr[i];      }        // printing values using pointer      for (i = 0; i < SIZE; i++) {            printf("Value of arr[%d] = %d\n", i, \*ptr[i]);      }  } |   **Output:**  Value of arr[0] = 1  Value of arr[1] = 2  Value of arr[2] = 3  **Example:** We can likewise make an array of pointers to the character to store a list of strings.  filter\_none  edit  play\_arrow  brightness\_4   |  | | --- | | #include <stdio.h>    const int size = 4;    void main()  {        // array of pointers to a character      // to store a list of strings      char\* names[] = {          "amit",          "amar",          "ankit",          "akhil"      };        int i = 0;        for (i = 0; i < size; i++) {          printf("%s\n", names[i]);      }  } |   **Output:**  amit  amar  ankit  akhil |

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **23-June-2020** | **Name:** | **Raziya Banu** |
| **Course:** | **C++ Programming** | **USN:** | **4AL16EC058** |
| **Topic:** | **Functions** | **Semester & Section:** | **8th sem & ‘B’ section** |
| **AFTERNOON SESSION DETAILS** | | | |
| **Image of session** | | | |
| **C++ Functions**  A function is a block of code which only runs when it is called.  You can pass data, known as parameters, into a function.  Functions are used to perform certain actions, and they are important for reusing code: Define the code once, and use it many times. Create a Function C++ provides some pre-defined functions, such as main(), which is used to execute code. But you can also create your own functions to perform certain actions.  To create (often referred to as declare) a function, specify the name of the function, followed by parentheses **()**: Syntax void myFunction() {   // code to be executed } **Example Explained**  * myFunction() is the name of the function * void means that the function does not have a return value. You will learn more about return values later in the next chapter * inside the function (the body), add code that defines what the function should do  Call a Function Declared functions are not executed immediately. They are "saved for later use", and will be executed later, when they are called.  To call a function, write the function's name followed by two parentheses () and a semicolon ;  In the following example, myFunction() is used to print a text (the action), when it is called: Example Inside main, call myFunction():  // Create a function void myFunction() {   cout << "I just got executed!"; }  int main() {   **myFunction();** // call the function   return 0; }  // Outputs "I just got executed!"  A function can be called multiple times: Example void myFunction() {   cout << "I just got executed!\n"; }  int main() {   **myFunction();**   **myFunction();**   **myFunction();**   return 0; }  // I just got executed! // I just got executed! // I just got executed! Function Declaration and Definition A C++ function consist of two parts:   * **Declaration:** the function's name, return type, and parameters (if any) * **Definition:** the body of the function (code to be executed)   void **myFunction()** { // **declaration**   // the body of the function (**definition**) }  **Note:** If a user-defined function, such as myFunction() is declared after the main() function, **an error will occur**. It is because C++ works from top to bottom; which means that if the function is not declared above main(), the program is unaware of it: Example int main() {   myFunction();   return 0; }  void myFunction() {   cout << "I just got executed!"; } Example // **Function declaration** void myFunction();  // The main method int main() {   myFunction();  // **call** the function   return 0; }  // **Function definition** void myFunction() {   cout << "I just got executed!"; } | | | |