**EXPERIMENT NO. 01**

|  |  |
| --- | --- |
| **DATE OF PERFORMANCE:** | **GRADE:** |
| **DATE OF ASSESSMENT:** | **SIGNATURE OF LECTURER/ TTA:** |

**AIM: Introduction to Data Structure.**

**THEORY:**

**Data structure is a representation of logical relationship existing between individual elements of data. In other words, a data structure defines a way of organizing all data items that considers not only the elements stored but also their relationship to each other. The term data structure is used to describe the way data is stored.**

**To develop a program of an algorithm we should select an appropriate data structure for that algorithm. Therefore, data structure is represented as:**

**Algorithm + Data structure = Program**

**Data structures are divided into two types:**

**• Primitive data structures.**

**• Non-primitive data structures.**

**Primitive Data Structures are the basic data structures that directly operate upon the machine instructions. They have different representations on different computers. Integers, floating point numbers, character constants, string constants and pointers come under this category.**

**Non-primitive data structures are more complicated data structures and are derived from primitive data structures. They emphasize on grouping same or different data items with relationship between each data item. Arrays, lists and files come under this category. Figure 1.1 shows the classification of data structures.**



***Figure 1.1: Introduction to Data Structure***

* **Dynamic memory allocation and deallocation in c:**

|  |  |
| --- | --- |
| **S.N.** | **Function & Description** |
| **1** | **(type\_cast\*) malloc (N \* sizeof(type));**  **This function allocates an array of num bytes and leave them initialized.** |
| **2** | **(type\_cast \*)calloc (n, size of (type));**  **This function allocates an array of num elements each of which size in bytes will be size.** |
| **3** | **realloc(var\_name, newsize);**  **This function re-allocates memory extending it upto newsize.** |
| **4** | **free(var\_name);**  **This function releases a block of memory block specified by address.** |

**PROGRAM -1: WRITE A PROGRAM TO SHOW USE OF MALLOC () AND FREE() FUNCTION.**

**#include <stdio.h>**

**#include <stdlib.h> // Corrected: Removed unnecessary conio.h**

**int main() { // Corrected: Changed "Void" to "int" for the return type of main**

**int i, N;**

**float \*height;**

**float sum = 0, avg;**

**printf("Input no. of students\n");**

**scanf("%d", &N);**

**height = (float \*)malloc(N \* sizeof(float));**

**if (height == NULL) {**

**printf("Memory allocation failed. Exiting the program.\n");**

**return 1; // Exiting with an error status**

**}**

**printf("Input heights for %d students \n", N);**

**for (i = 0; i < N; i++)**

**scanf("%f", &height[i]);**

**for (i = 0; i < N; i++)**

**sum += height[i];**

**avg = sum / (float)N;**

**printf("Average height = %f \n", avg);**

**free(height);**

**return 0;**

**}**

**OUTPUT:**

**Input no. of students**

**5**

**Input heights for 5 students**

**165.2**

**172.8**

**158.9**

**170.5**

**180.0**

**Average height = 169.480011**

**PROGRAM-2: WRITE A PROGRAM TO SHOW USE OF CALLOC () AND FREE() FUNCTION.**

**#include <stdio.h>**

**#include <stdlib.h>**

**int main()**

**{**

**int i, n;**

**int \*a;**

**printf("Number of elements to be entered: ");**

**scanf("%d", &n);**

**a = (int \*)calloc(n, sizeof(int));**

**if (a == NULL)**

**{**

**printf("Memory allocation failed. Exiting the program.\n");**

**return 1;**

**}**

**printf("Enter %d numbers:\n", n);**

**for (i = 0; i < n; i++)**

**{**

**scanf("%d", &a[i]);**

**}**

**printf("The numbers entered are: ");**

**for (i = 0; i < n; i++)**

**{**

**printf("%d ", a[i]);**

**}**

**free(a);**

**return 0;**

**}**

**OUTPUT:**

**Number of elements to be entered: 5**

**Enter 5 numbers:**

**11**

**22**

**33**

**44**

**55**

**The numbers entered are: 11 22 33 44 55**

**PROGRAM-3: WRITE A PROGRAM TO SHOW USE OF REALLOC () FUNCTION.**

**#include <stdio.h>**

**#include <stdlib.h>**

**int main() {**

**int \*ptr, i, n1, n2;**

**printf("Enter size of array: ");**

**scanf("%d", &n1);**

**ptr = (int\*)malloc(n1 \* sizeof(int));**

**if (ptr == NULL) {**

**printf("Memory allocation failed. Exiting the program.\n");**

**return 1;**

**}**

**printf("Address of previously allocated memory: ");**

**for (i = 0; i < n1; ++i) {**

**printf("%p\t", (ptr + i));**

**}**

**printf("\nEnter new size of array: ");**

**scanf("%d", &n2);**

**ptr = (int\*)realloc(ptr, n2 \* sizeof(int));**

**if (ptr == NULL) {**

**printf("Memory reallocation failed. Exiting the program.\n");**

**return 1;**

**}**

**printf("Address of reallocated memory: ");**

**for (i = 0; i < n2; ++i) {**

**printf("%p\t", (ptr + i));**

**}**

**free(ptr);**

**return 0;**

**}**

**OUTPUT:**

Enter size of array: 3

Address of previously allocated memory: 0x7f98c3403010 0x7f98c3403014 0x7f98c3403018

Enter new size of array: 5

Address of reallocated memory: 0x7f98c3403010 0x7f98c3403014 0x7f98c3403018 0x7f98c340301c 0x7f98c3403020