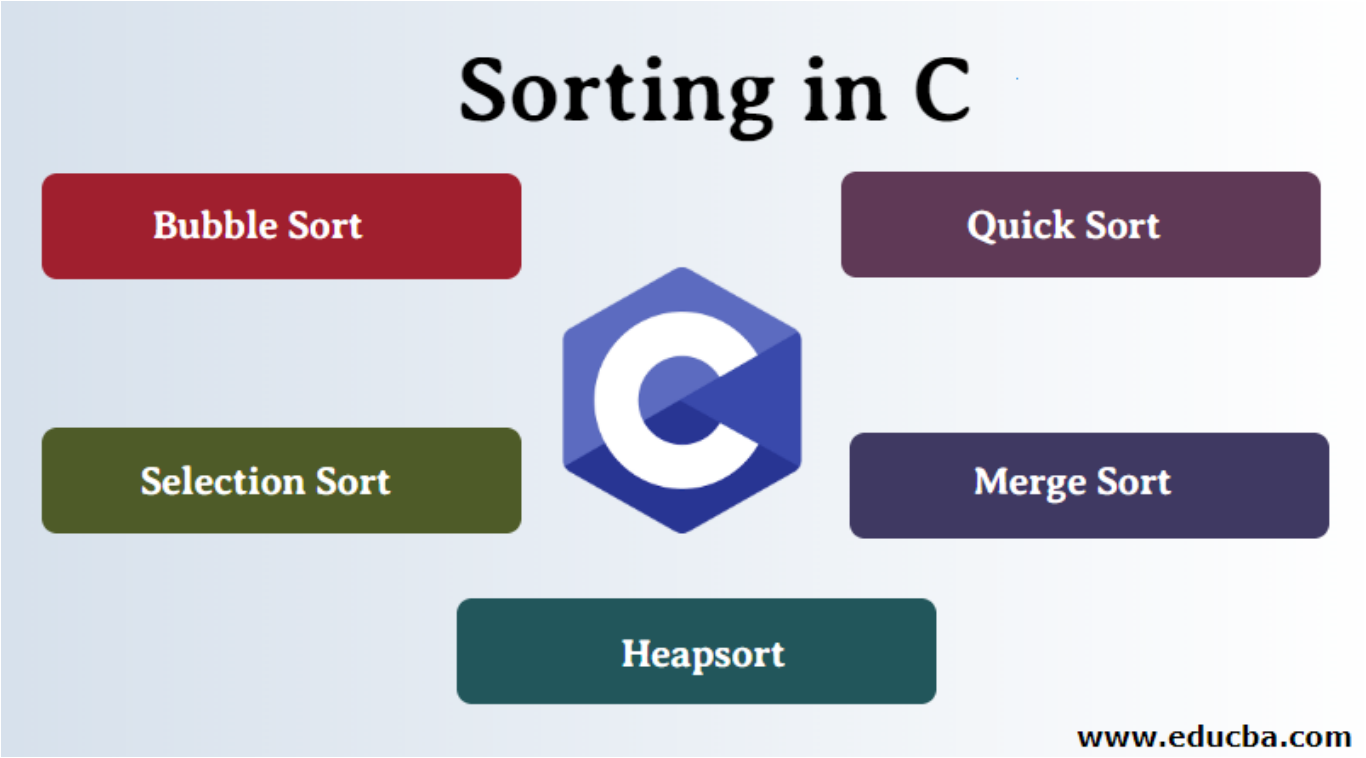
EDUCBA
(https://www.educba.com/software-development/)

← (https://www.educba.com/strings-array-in-c/)

→ (https://www.educba.com/heap-sort-in-c/)



Introduction to Sorting in C

QUIZ

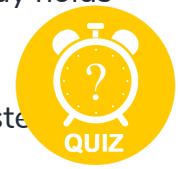
The process of Sorting can be explained as a technique of rearranging the elements in a particular order, which can be set ready for further processing by the program logic. In C programming language, there are multiple sorting algorithms available, which can be

(https://www.educba.com/software-development/)

Let us learn how sorting is performed in C:

- Sorting can be performed in various ways based on the sorting algorithm. In C programming language we do have several approaches to sort the list. The term sorting states arranging of data in a particular manner usually in ascending order. Though the way to sort the data is different in all of the sorting algorithms (https://www.educba.com/sorting-algorithms-in-javascript/), the outcome of all of them is the same.
- Usually, in sorting, the program searches for the minimum number and shifted that number to the beginning of the list and repeat the same searches. Again once the other small number is encountered, it is shifted to the next space in the list right after the first index and this process keeps on repeating until the sort list is obtained. This is the way sorting is done in the C programming language.
- In all the approaches to sort the list, the array plays a very vital role in the C programming language. In every algorithm, the array has been used to store the list of the elements that have to be sorted. For instance, in bubble sort, the elements are stored in the single array and the values in the array have been processed to convert them into a list of sorted data.
- In the selection sort, the same array has been treated as two arrays where the first array is considered to be vacant in order to tell the sorted values while the second array holds

  the unsorted list. To serve the purpose of sorting the array is used very often inste holding the values in individual variables. Among all of the algorithms, quick sort works very quick and hence named quick sort. It takes much less time as compared to the other



(https://www.educba.com/software-development/)

1. Bubble Sort

- Bubble sort may be defined as the sorting algorithm that follows the approach of replacing the value in the first index with the smallest value in the array and keep it repeating until the list is sorted. It is a very simple way of performing sorting. In this way to sort the array, the value has to be assigned to the array in the beginning before starting the sorting.
- Below is the program to sort the array using bubble sort where the values have been taken from the user. Once the program is compiled and run, it will ask the user for the number of elements that they want to sort. Once the number is provided, the program will ask the user to provide values equivalent to the count that they have provided. The values will be stored in the array and will be processed further using nested for loop together with decision making using “if” in order to sort the array.
- The first smallest value found in the array has been moved to the first index of the array and then the search begins again to find the other smallest number. Once the next smallest number is found, it replaces the value in the second index and the process keeps on repeating until the array consists of a sorted list of values.

Code:

```
#include <stdio.h>
int main()
{

int total_count, counter, counter1, swap_var;
int array[20];
printf("How many number you want to input?\n");
```

(https://www.educba.com/software-development/)

```
scanf( "%d" , &array[counter]);
for (counter = 0 ; counter < total_count - 1; counter++)
{
for (counter1 = 0 ; counter1 < total_count - counter - 1;
counter1++)
{
if (array[counter1] > array[counter1+1]) /* For decreasing order
use < */
{
swap_var        = array[counter1];
array[counter1]   = array[counter1+1];
array[counter1+1] = swap_var;
}
}
}
printf("Below is the list of elements sorted in ascending
order:\n");
for (counter = 0; counter < total_count; counter++)
printf("%d\n", array[counter]);
return 0;
}
```

The user has submitted the input 5 3 60 14 1 2 645. The algorithm has been applied on array consisting of values in the manner it is provided by the user and after processing it the output we received is 1 2 3 5 14 60 645.

(https://www.educba.com/software-development/)

```
66
14
1
2
645
Below is the list of elements sorted in ascending order:
1
2
3
5
14
66
645
```

2. Selection Sort

- The selection sort may be defined as another algorithm for sorting the list in which the array is bifurcated into two arrays where the first array is supposed to be empty while the second array consists of the unsorted list of values. The program searches for the smallest values in the second array and when the value is found, it has been moved to the beginning of the first array that was empty. The approach is repeated again and the next smallest values will be shifted to the second index of the first array. The processes will keep on repeating until the second array became empty.
- The below program is the coding implementation of the selection sort algorithm. Once the program runs successfully, it will request the user to input the count of values that they are willing to sort. Once the count is obtained, the program will ask the user to input the values for the array that has to be sorted. The value is then processed using nested for loop in order to sort the numbers. The if condition checking has also been involved here to check the smallest number.

- The processes will be repeated until the first list is full of the sorted list. Meanwhil programs keep its primary focus to check if the second array is having value and if it is found positive, the program runs the sorting algorithm again. Though it sorts the list in

(https://www.educba.com/software-development/)

```
#include<stdio.h>
#include<conio.h>
void main()
{
int total_count,counter1,counter2,minimum,temp_value;
int a[20];
printf("\n Enter the Number of Elements: ");
scanf("%d",&total_count);
printf("\n Enter %d Elements: ",total_count);
for(counter1=0;counter1<total_count;counter1++)
{
scanf("%d",&a[counter1]);
}
for(counter1=0;counter1<total_count-1;counter1++)
{
minimum=counter1;
for(counter2=counter1+1;counter2<total_count;counter2++)
{
if(a[minimum]>a[counter2])
minimum=counter2;
}
if(minimum!=counter1)

{
temp_value=a[counter1];
a[counter1]=a[minimum];
```

(https://www.educba.com/software-development/)

```
for(counter1=0;counter1<total_count;counter1++)
{
printf("%d ",a[counter1]);
}
getch();
}
```

On asking for the count of elements that has to be sorted, the user has provided 6 in the below output. Later the values that have been input are 25 65 36 86 96 45. These values are stored in the array which is expected to be bifurcated into two arrays where one will be empty to store the sorted list and the other will be having the unsorted list. After processing the input, the outcome was 25 36 45 65 86 96. This lost has been sorted using the selection sort. Once all the six values have been moved to the first array in the sorted form, the second array will become empty and the algorithm will be terminated.

Output:

```
Please enter the count of elements you want to sort: 6

Please enter 6 Elements: 25
65
```

(https://www.educba.com/software-development/)

3. Quick Sort

- Quicksort can be defined as the other algorithm for sorting the list in which the approach is to divide the array in terms of greater than and less than values until the entire values if divided into individuals forms. In this algorithm, the value of the last index of the array has been selected as a pivot and all the values smaller than pivot has been shifted to the array that is expected to occur in the left of the value and the elements having a higher value than the pivot are shifted to the right array. Again one pivot is selected from the newly formed array that had the values less than the last pivot value. Similarly, the values smaller than the new pivot will be shifted to the array that will be left and the values more than the new pivot will be shifted in the right array.
- The below program is the quicksort implementation using the C programming language. Once the program runs, it will ask the user for the number of elements that they want to sort. Based on the count, the for loop will iterate estimated times to take the input from the user. The input will be processed using the if conditions together with the for loop in order to generate a sorted list. The array will keep on arranging the values using the pivot value until all the values have been checked for the smallest value.
- The sorting done using (https://www.educba.com/sorting-in-c-sharp/) this algorithm is way too faster as compared to the other sorting algorithms and that's why it has been named quick sort. Quicksort is the only algorithm that leads to dividing the array until all the values are separated into the individual arrays. They will be then added or aggregated in a single array which is considered as the sorted list.

Code:

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

(https://www.educba.com/software-development/)

```
printf( Please enter the total count of the elements that you want
to sort: ");
scanf("%d", &count);
printf("Please input the elements that has to be sorted:\n");
for (counter = 0; counter < count; counter++)
{
scanf("%d", &element_list[counter]);
}
quicksort_method(element_list, 0, count - 1);
printf("Output generated after using quick sort\n");
for (counter = 0; counter < count; counter++)
{
printf("%d ", element_list[counter]);
}
printf("\n");
return 0;
}
void quicksort_method(int element_list[], int low, int high)
{
int pivot, value1, value2, temp;
if (low < high)
{
pivot = low;
value1 = low;
value2 = high;
while (value1 < value2)
```

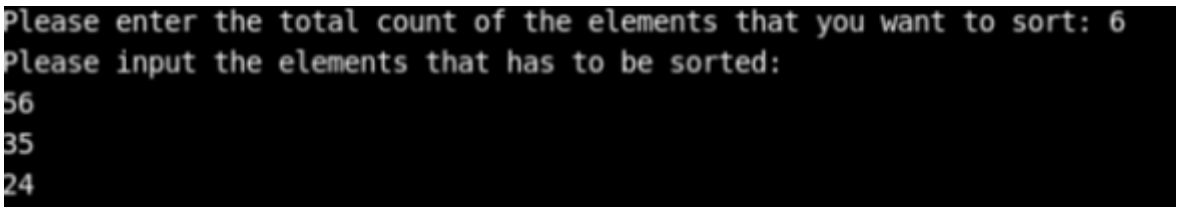
EDUCBA
(https://www.educba.com/software-development/)

```
value1++;
}
while (element_list[value2] > element_list[pivot] && value2 >=
low)
{
value2--;
}
if (value1 < value2)
{
temp = element_list[value1];
element_list[value1] = element_list[value2];
element_list[value2] = temp;
}
}
temp = element_list[value2];
element_list[value2] = element_list[pivot];
element_list[pivot] = temp;
quicksort_method(element_list, low, value2 - 1);
quicksort_method(element_list, value2 + 1, high);
}
}
```

⚡ Popular Course in this category

In the below output, the user confirmed that they will be submitting 6 values and to form a list of sorted data. After providing the count, the values provided by the user are 56, 35, 24, 86, 98, 2. The quicksort has been applied to these values and the sorted list has been generated that has the value 2, 24, 35, 56,86,98.

Output:



(https://www.educba.com/software-development/)

4. Merge Sort

- Merge sort may be defined as another sorting algorithm that performs the sorting by segregating the array till last when it turns into an individual value and then aggregating them in a manner so that it could turn into a sorted array.
- The process consumes a bit much time as compared to the other rival algorithms but it is considered pretty efficient as compared to others. When it comes to sorting a large list, this algorithm works very fine and hence preferred in developing the application that has to process the large list.

Code:

```
#include<stdio.h>
void algo_merge_sort(int val[],int counter1,int counter2);
void perfrom_merge(int val[],int counter11,int counter12,int
counter22,int counter21);
int main()
{
int val[100],chk,counter1;
printf("Please enter the total count of the elements that you want
to sort: \n");
scanf("%d",&chk);
printf("Please input the elements that has to be sorted:\n");
for(counter1=0;counter1<chk;counter1++)
scanf("%d",&val[counter1]);
algo_merge_sort(val,0,chk-1);
printf("\n Output generated after using quick sort \n");
```

(https://www.educba.com/software-development/)

```
void algo_merge_sort(int val[],int counter1,int counter2)
{
int mid;
if(counter1<counter2)
{
mid=(counter1+counter2)/2;
algo_merge_sort(val,counter1,mid);
algo_merge_sort(val,mid+1,counter2);
perfrom_merge(val,counter1,mid,mid+1,counter2);
}
}
void perfrom_merge(int val[],int counter11,int counter12,int
counter22,int counter21)
{
int temp_val[50];
int c1,c2,c3;
c1=counter11;
c2=counter22;
c3=0;
while(c1<=counter12 && c2<=counter21)
{
if(val[c1]<val[c2])
temp_val[c3++]=val[c1++];
else
temp_val[c3++]=val[c2++];
}
```

(https://www.educba.com/software-development/)

```
for(c1=counter11,c2=0,c1<counter21,c1++,c2++)
val[c1]=temp_val[c2];
}
```

When the above code runs, it first asks the user to provide the number of elements that they want to sort. Once the number has been submitted, they will need to provide the values of equal count that they have provided initially. Once the values have been submitted, the algorithm will hold those values in the array and will process it to transform the array into the sorted array. After the array is sorted in ascending order, the output will be displayed to the user.

Output:

```
Please enter the total count of the elements that you want to sort:
6
Please input the elements that has to be sorted:
32
36
65
98
45
8

 Output generated after using quick sort
8 32 36 45 65 98
```

5. Heapsort

- The heap sort (https://www.educba.com/heap-sort-in-c/) can be defined as the so algorithm that works by searching the maximum element in the list and place it to the last. The algorithm performs the action recursively until the array gets sorted into the

(https://www.educba.com/software-development/)

the implementation of this algorithm in the C programming language together with the output.

Code:

```
#include<stdio.h>
void form(int []);
void set_down(int [],int);
int main()
{
int val[100],chk,counter,end,temp_val;
printf("Please enter the total count of the elements that you want to sort: \n");
scanf("%d",&chk);
printf("Please input the elements that has to be sorted:\n");
for(counter=1;counter<=chk;counter++)
scanf("%d",&val[counter]);
val[0]=chk;
form(val);
while(val[0] > 1)
{
end=val[0];
temp_val=val[1];

val[1]=val[end];
val[end]=temp_val;
val[0]--;
```

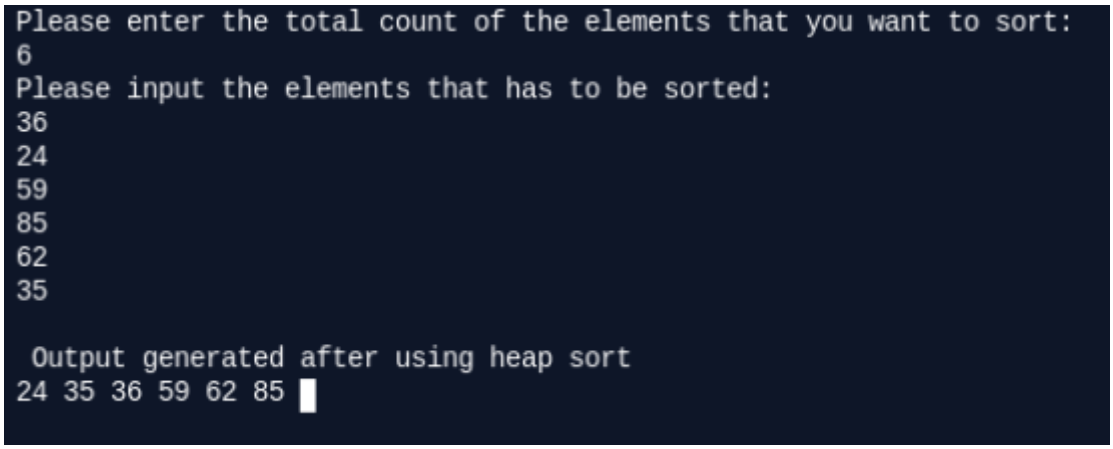
(https://www.educba.com/software-development/)

```
printf("%d ",val[counter]);
}
void form(int val[])
{
int counter,chk;
chk=val[0];
for(counter=chk/2;counter>=1;counter--)
set_down(val,counter);
}
void set_down(int val[],int counter)
{
int counter2,temp_val,chk,flag=1;
chk=val[0];
while(2*counter<=chk && flag==1)
{
counter2=2*counter;
if(counter2+1<=chk && val[counter2+1] > val[counter2])
counter2=counter2+1;
if(val[counter] > val[counter2])
flag=0;
else
{
temp_val=val[counter];
val[counter]=val[counter2];
val[counter2]=temp_val;
counter=counter2;
```

(https://www.educba.com/software-development/)

The working of this algorithm is the same as that of other sorting algorithms as it also sorts the list in ascending order. When the above-written code runs, the user has to submit the count of values that they will be sorting. Once the values are submitted, the code will process them in order to turn the array into the sorted one. The output will be shown eventually and it can be observed that the values that have been submitted by the user has sorted in ascending order.

Output:



6. Insertion Sort

- Insertion sort (https://www.educba.com/insertion-sort-in-javascript/) may be defined as the sorting algorithm that works by moving the minimum value at the beginning of the list one at a time. This is a very less efficient sorting algorithm and not found suitable to deal with the large list.
- This approach of sorting the algorithm works very slowly and usually not preferred in any of the applications. It can work good with the list that has pretty few numbers of

  elements. For the applications, that have the requirement to process a few numbe values can leverage this algorithm.

(https://www.educba.com/software-development/)

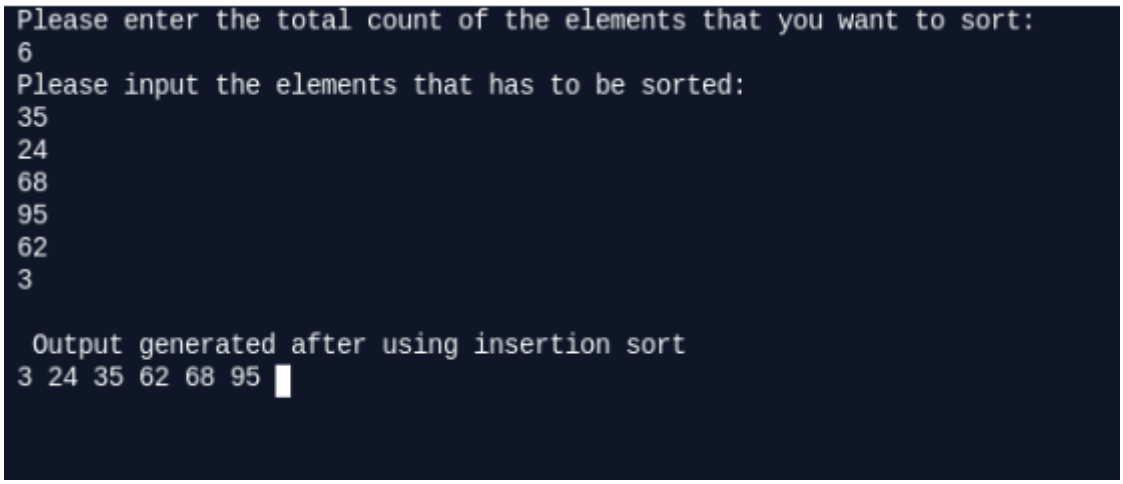
```
int counter1,counter2,chk,temp_val,val[100];
printf("Please enter the total count of the elements that you want to sort: \n");
scanf("%d",&chk);
printf("Please input the elements that has to be sorted:\n");
for(counter1=0;counter1<chk;counter1++)
{
scanf("%d",&val[counter1]);
}
for(counter1=1;counter1<=chk-1;counter1++)
{
temp_val=val[counter1];
counter2=counter1-1;
while((temp_val<val[counter2])&&(counter2>=0))
{
val[counter2+1]=val[counter2];
counter2=counter2-1;
}
val[counter2+1]=temp_val;
}
printf("\n Output generated after using insertion sort \n");
for(counter1=0;counter1<chk;counter1++)

{
printf("%d ",val[counter1]);
}
```

(https://www.educba.com/software-development/)

Afterward, the values entered by the user will be stored into the array. They will then go under processing and by the use of for loop and condition checking, the minimum value will be moved to the beginning in every recursion and end up by generating a sorted array. The values will be displayed to the user at the end of the program.

Output:



Conclusion

The sorting algorithm is used to generate a sorted list which is a normal list where all the values are sorted in a particular manner. The list has been used very often in the actual application to bring some functionalities. In this article we have covered bubble sort, selection sort, and quicksort while there are several other algorithms like merge sort are also there that can be leveraged to generate a sorted list. Among all of the sorting algorithms, quicksort works very fast and helps to sort the list very quickly. The programs written here are basically to implement these sorting algorithms using the C programming language. If you are willing to imple same in other programming languages, you can use the same logic and the only thing that may vary can be the syntax and keywords.

(https://www.educba.com/software-development/)

more –

1. Patterns in C Programming (https://www.educba.com/patterns-in-c-programming/)
2. Palindrome in C Program (https://www.educba.com/palindrome-in-c-program/)
3. Merge Sort In Java (https://www.educba.com/merge-sort-in-java/)
4. Sorting in C++ (https://www.educba.com/sorting-in-c-plus-plus/)

EDUCBA
(https://www.educba.com/software-development/)

Sign Up (https://www.educba.com/software-development/signup/?source=footer)

Corporate Training (https://www.educba.com/corporate/?source=footer)

Certificate from Top Institutions (https://www.educba.com/educbalive/?source=footer)

Contact Us (https://www.educba.com/contact-us/?source=footer)

Verifiable Certificate (https://www.educba.com/software-development/verifiable-certificate/?source=footer)

Reviews (https://www.educba.com/software-development/reviews/?source=footer)

Terms and Conditions (https://www.educba.com/terms-and-conditions/?source=footer)

Privacy Policy (https://www.educba.com/privacy-policy/?source=footer)

Apps

iPhone & iPad (https://itunes.apple.com/in/app/educba-learning-app/id1341654580?mt=8)

Android (https://play.google.com/store/apps/details?id=com.educba.www)

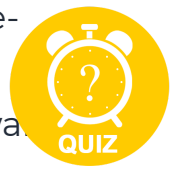
Resources

Free Courses (https://www.educba.com/software-development/free-courses/?source=footer)

Java Tutorials (https://www.educba.com/software-development/software-development-tutorials/java-tutorial/?source=footer)

Python Tutorials (https://www.educba.com/software-development/software-development-tutorials/python-tutorial/?source=footer)

All Tutorials (https://www.educba.com/software-development/software-development-tutorials/?source=footer)



(https://www.educba.com/software-development/)

development-course/?source=footer)

Become a Python Developer (https://www.educba.com/software-development/courses/python-certification-course/?source=footer)

Java Course (https://www.educba.com/software-development/courses/java-course/?source=footer)

Become a Selenium Automation Tester (https://www.educba.com/software-development/courses/selenium-training-certification/?source=footer)

Become an IoT Developer (https://www.educba.com/software-development/courses/iot-course/?source=footer)

ASP.NET Course (https://www.educba.com/software-development/courses/asp-net-course/?source=footer)

VB.NET Course (https://www.educba.com/software-development/courses/vb-net-course/?source=footer)

PHP Course (https://www.educba.com/software-development/courses/php-course/?source=footer)