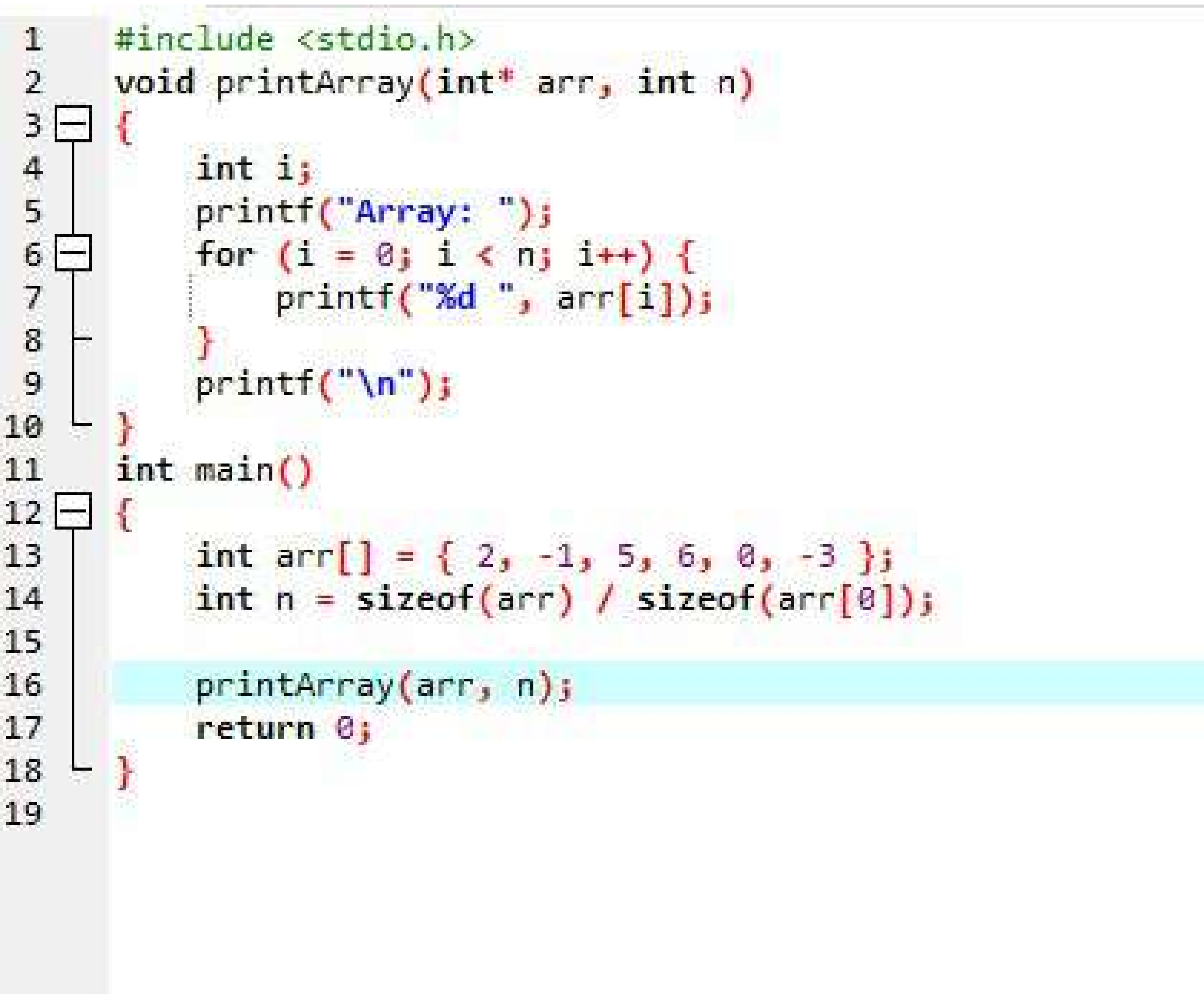
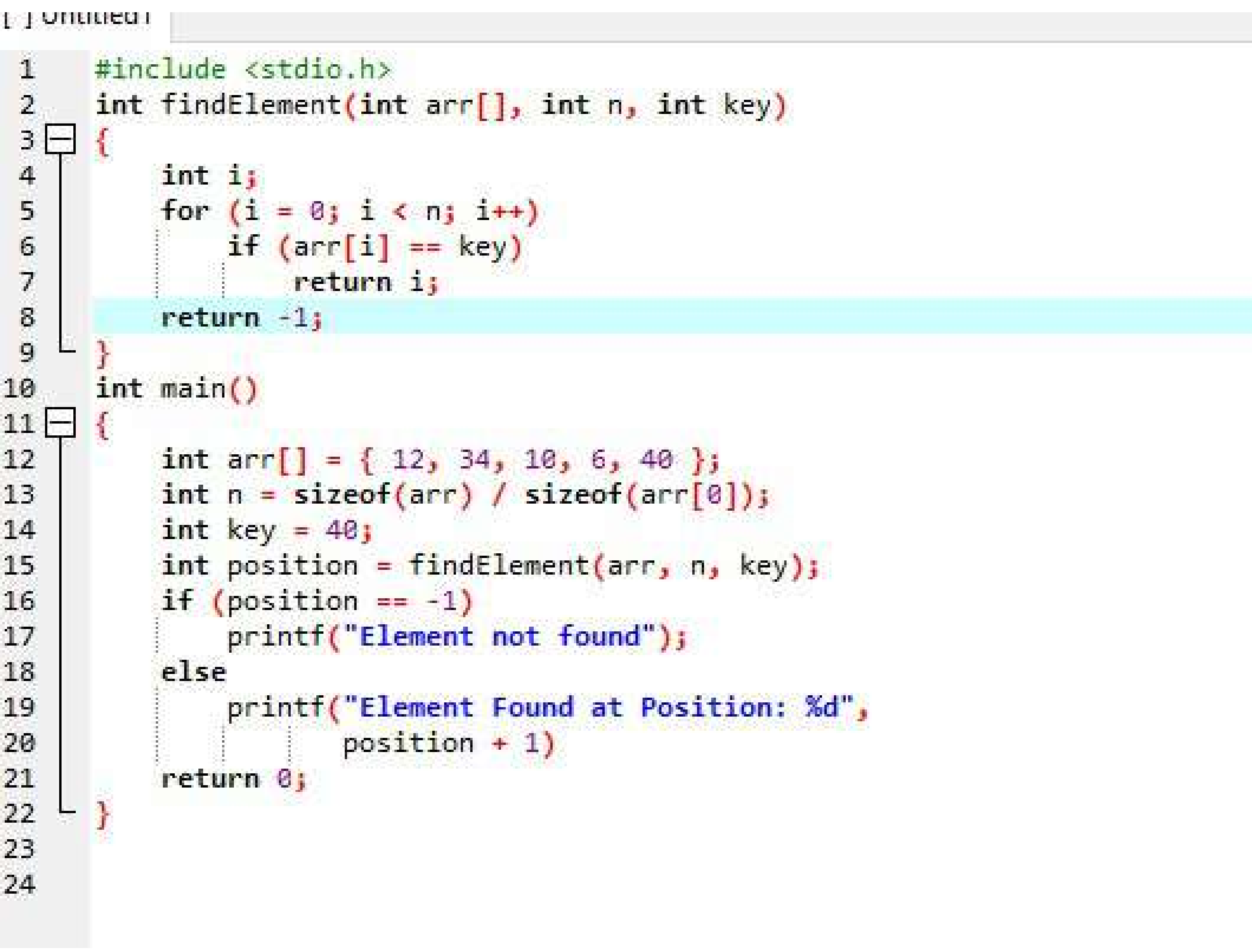
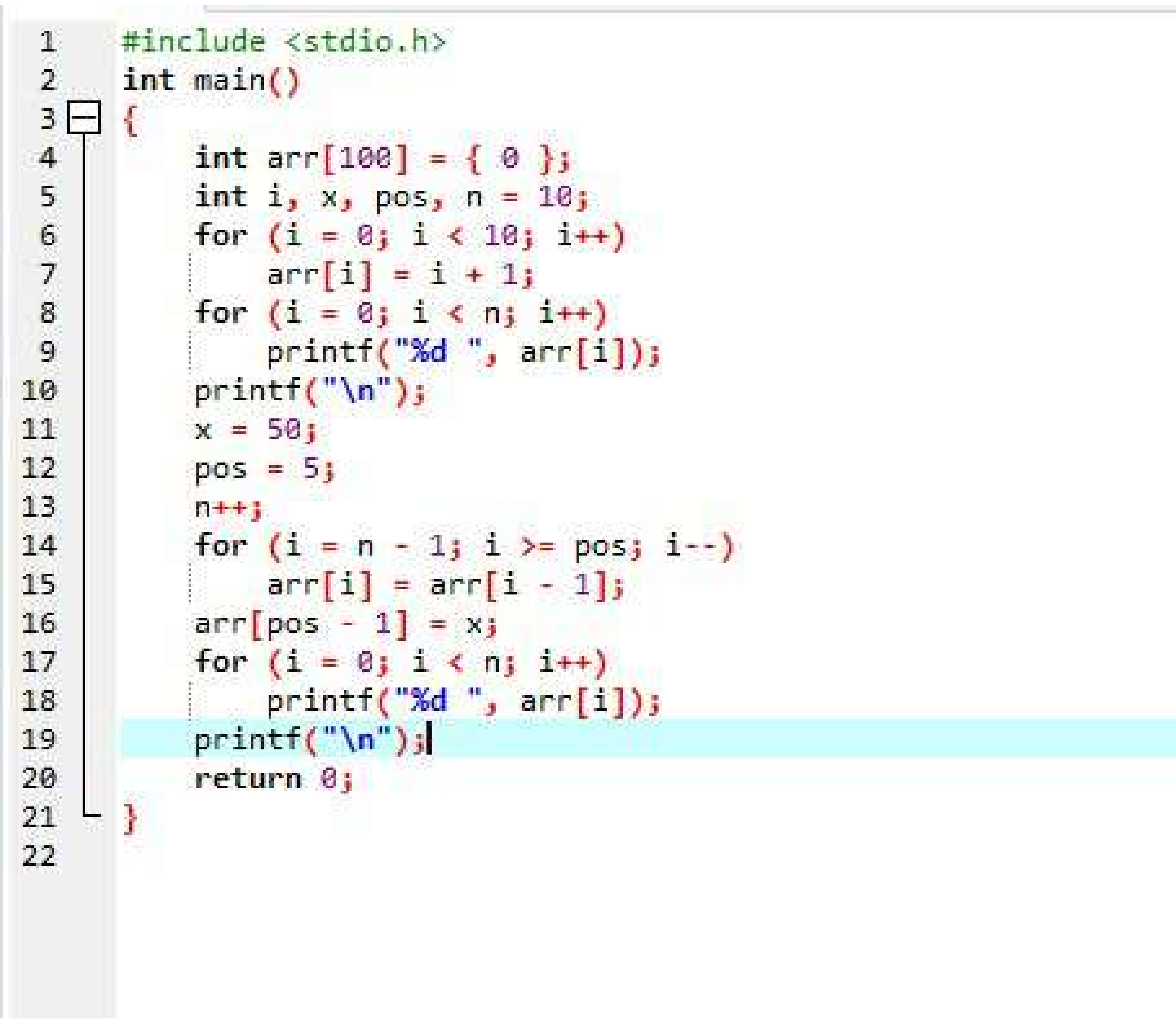


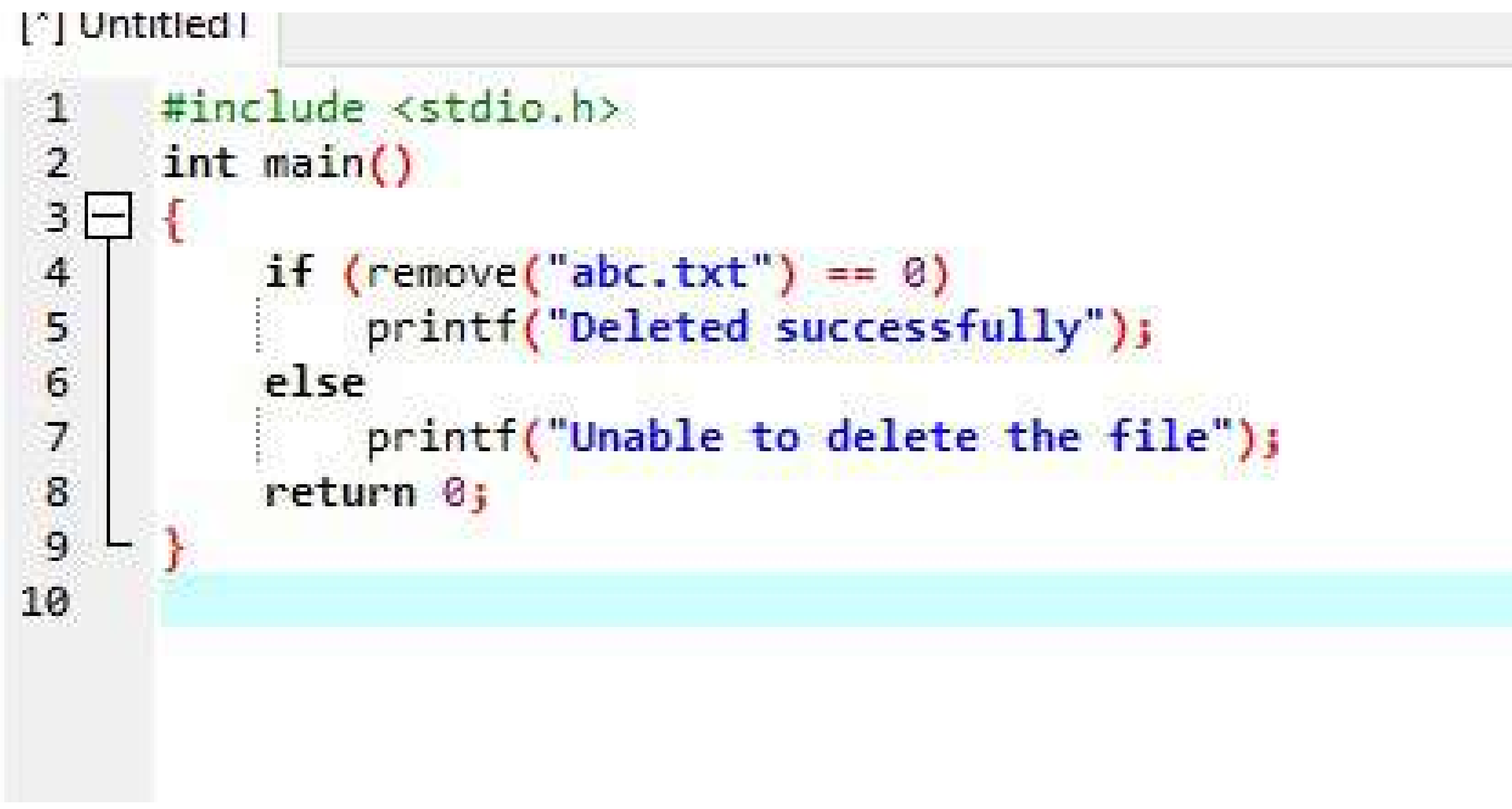
```c
#include <stdio.h>
void printArray(int* arr, int n)
{
    int i;
    printf("Array: ");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}
int main()
{
    int arr[] = { 2, -1, 5, 6, 0, -3 };
    int n = sizeof(arr) / sizeof(arr[0]);

    printArray(arr, n);
    return 0;
}
```



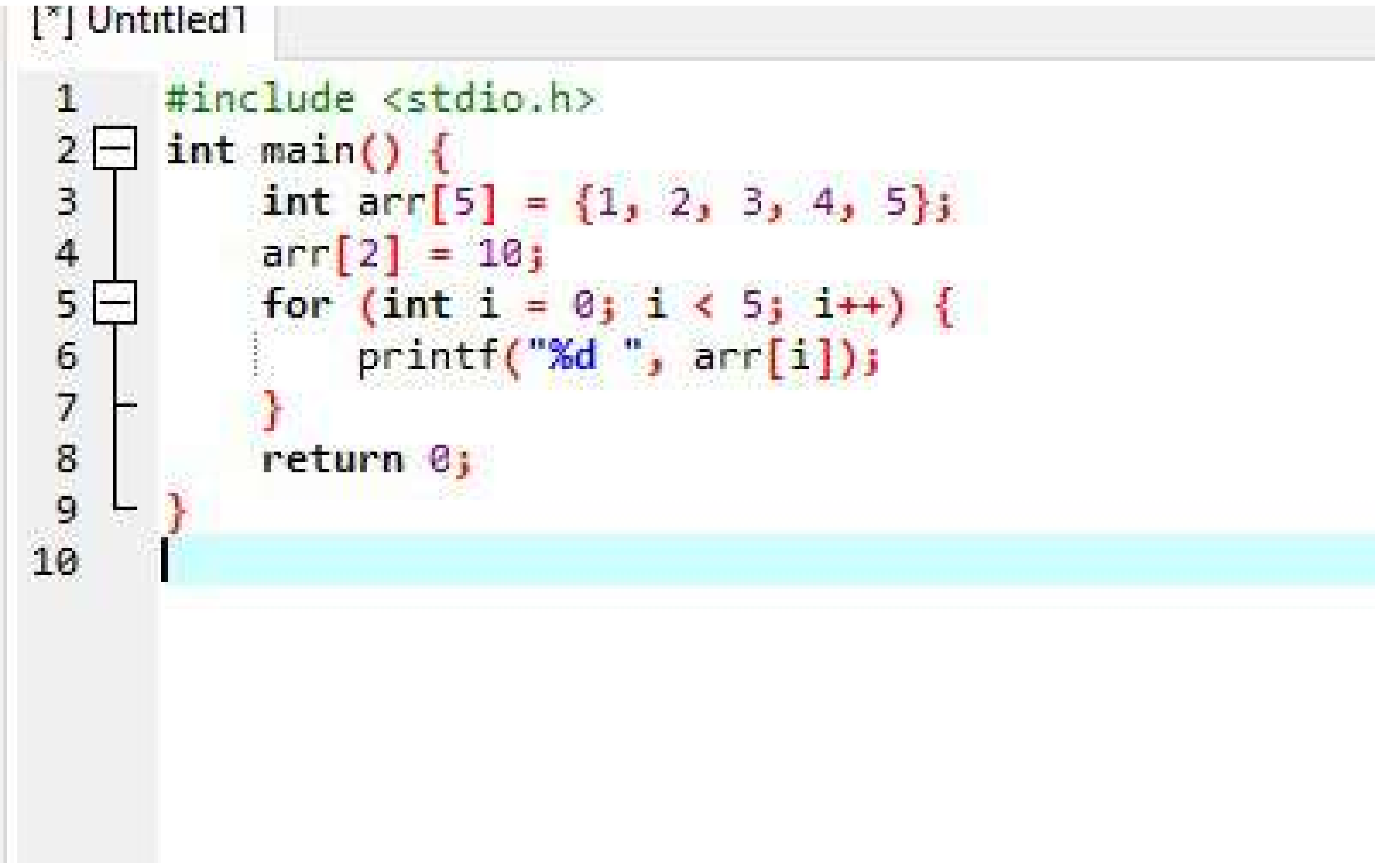
```c
#include <stdio.h>
int findElement(int arr[], int n, int key)
{
    int i;
    for (i = 0; i < n; i++)
        if (arr[i] == key)
            return i;
    return -1;
}
int main()
{
    int arr[] = { 12, 34, 10, 6, 40 };
    int n = sizeof(arr) / sizeof(arr[0]);
    int key = 40;
    int position = findElement(arr, n, key);
    if (position == -1)
        printf("Element not found");
    else
        printf("Element Found at Position: %d",
                  position + 1);
    return 0;
}
```



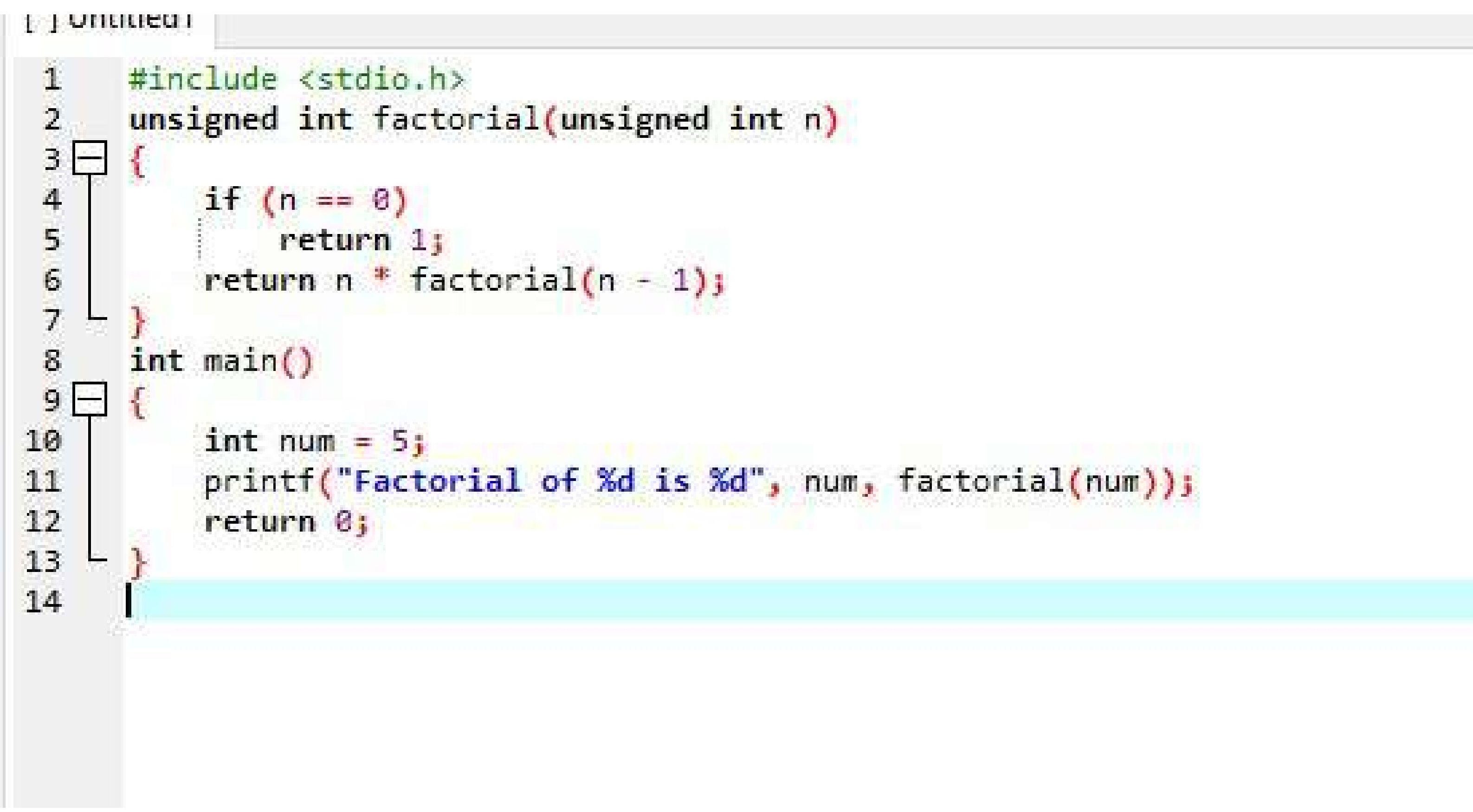
```c
#include <stdio.h>
int main()
{
    int arr[100] = { 0 };
    int i, x, pos, n = 10;
    for (i = 0; i < 10; i++)
        arr[i] = i + 1;
    for (i = 0; i < n; i++)
        printf("%d ", arr[i]);
    printf("\n");
    x = 50;
    pos = 5;
    n++;
    for (i = n - 1; i >= pos; i--)
        arr[i] = arr[i - 1];
    arr[pos - 1] = x;
    for (i = 0; i < n; i++)
        printf("%d ", arr[i]);
    printf("\n");
    return 0;
}
```



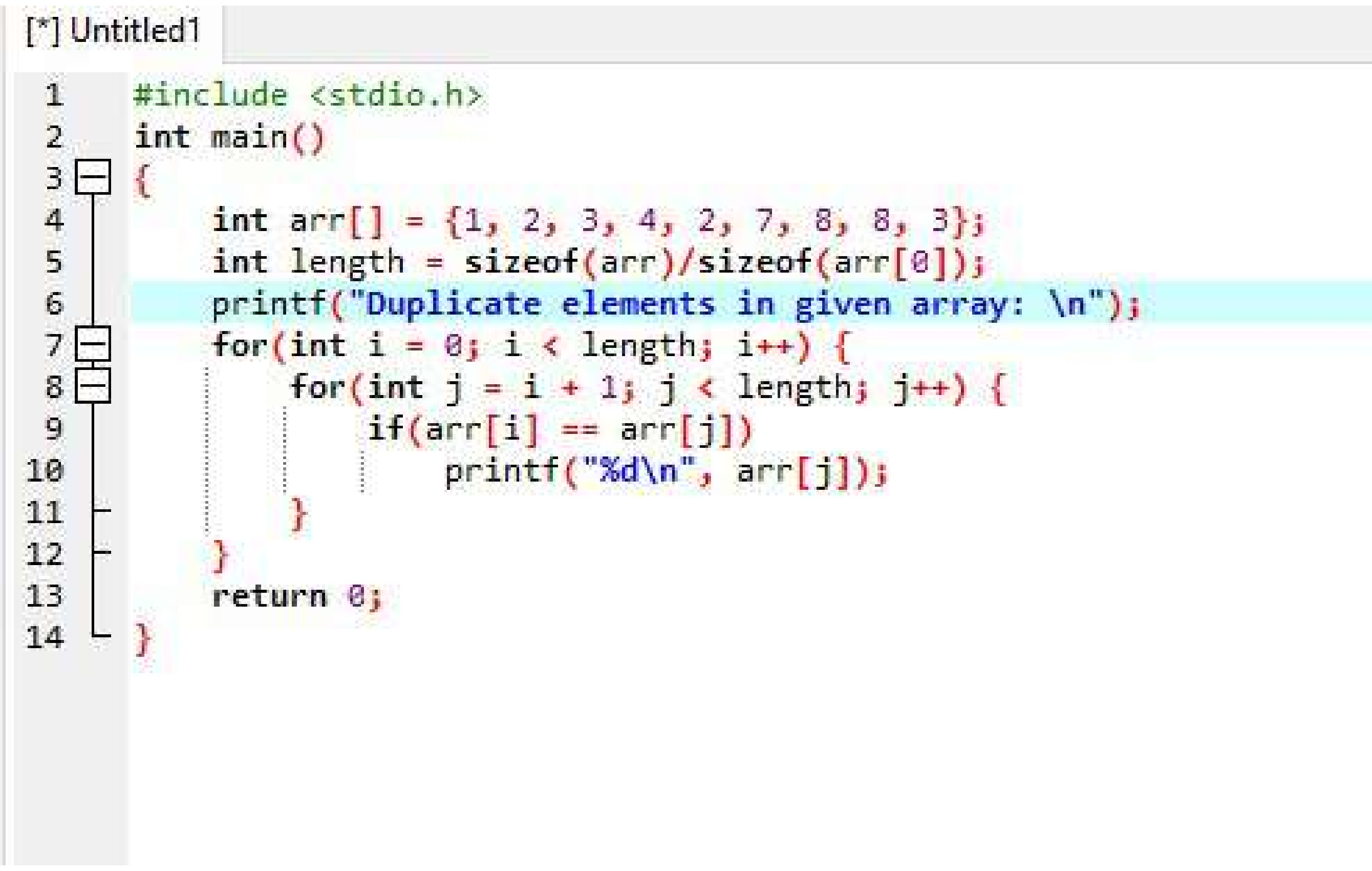
```
#include <stdio.h>
int main()
{
    if (remove("abc.txt") == 0)
        printf("Deleted successfully");
    else
        printf("Unable to delete the file");
    return 0;
}
```



```c
#include <stdio.h>
int main() {
	int arr[5] = {1, 2, 3, 4, 5};
	arr[2] = 10;
	for (int i = 0; i < 5; i++) {
		printf("%d ", arr[i]);
	}
	return 0;
}
```

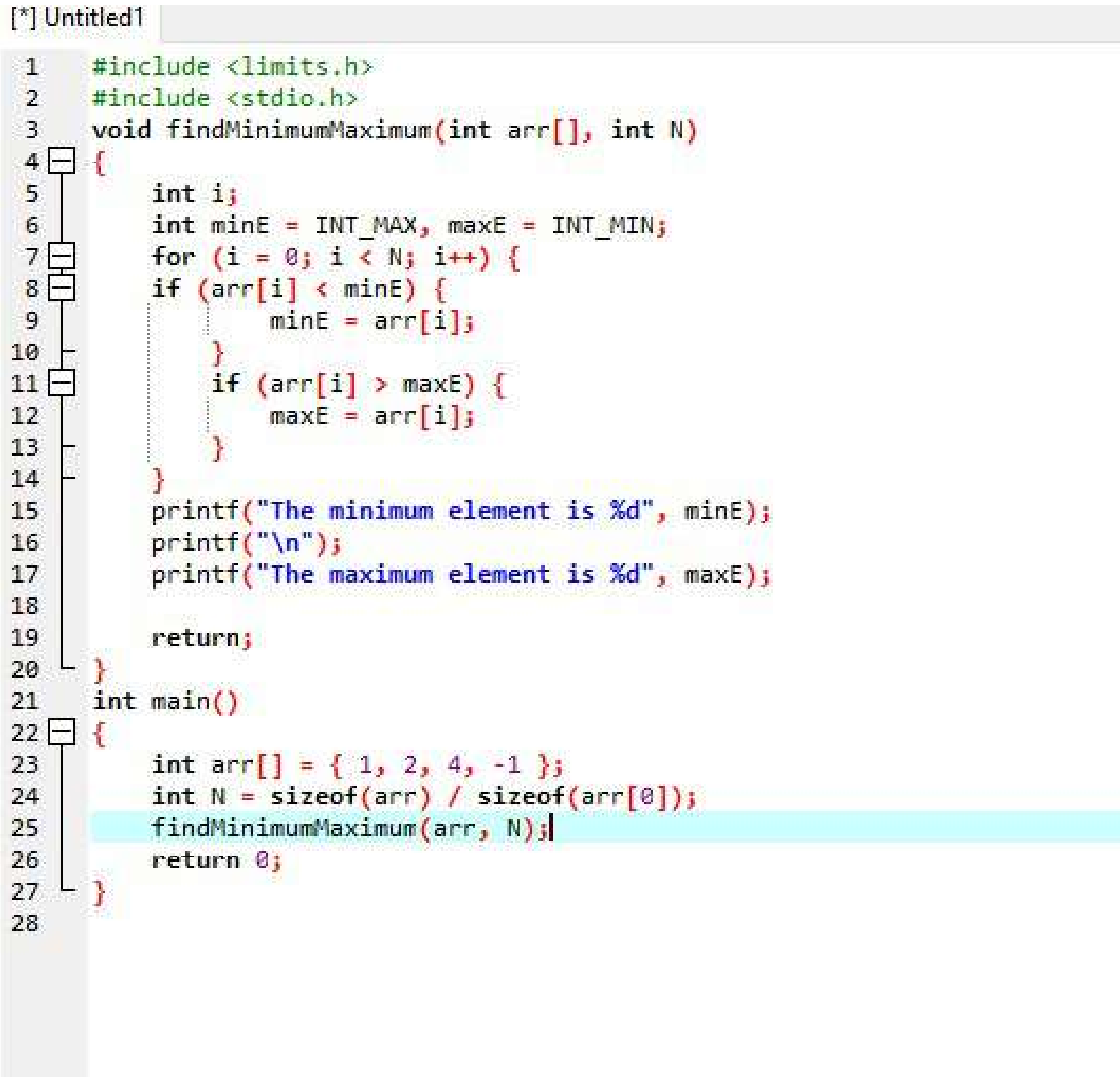


```c
#include <stdio.h>
unsigned int factorial(unsigned int n)
{
    if (n == 0)
        return 1;
    return n * factorial(n - 1);
}
int main()
{
    int num = 5;
    printf("Factorial of %d is %d", num, factorial(num));
    return 0;
}
```

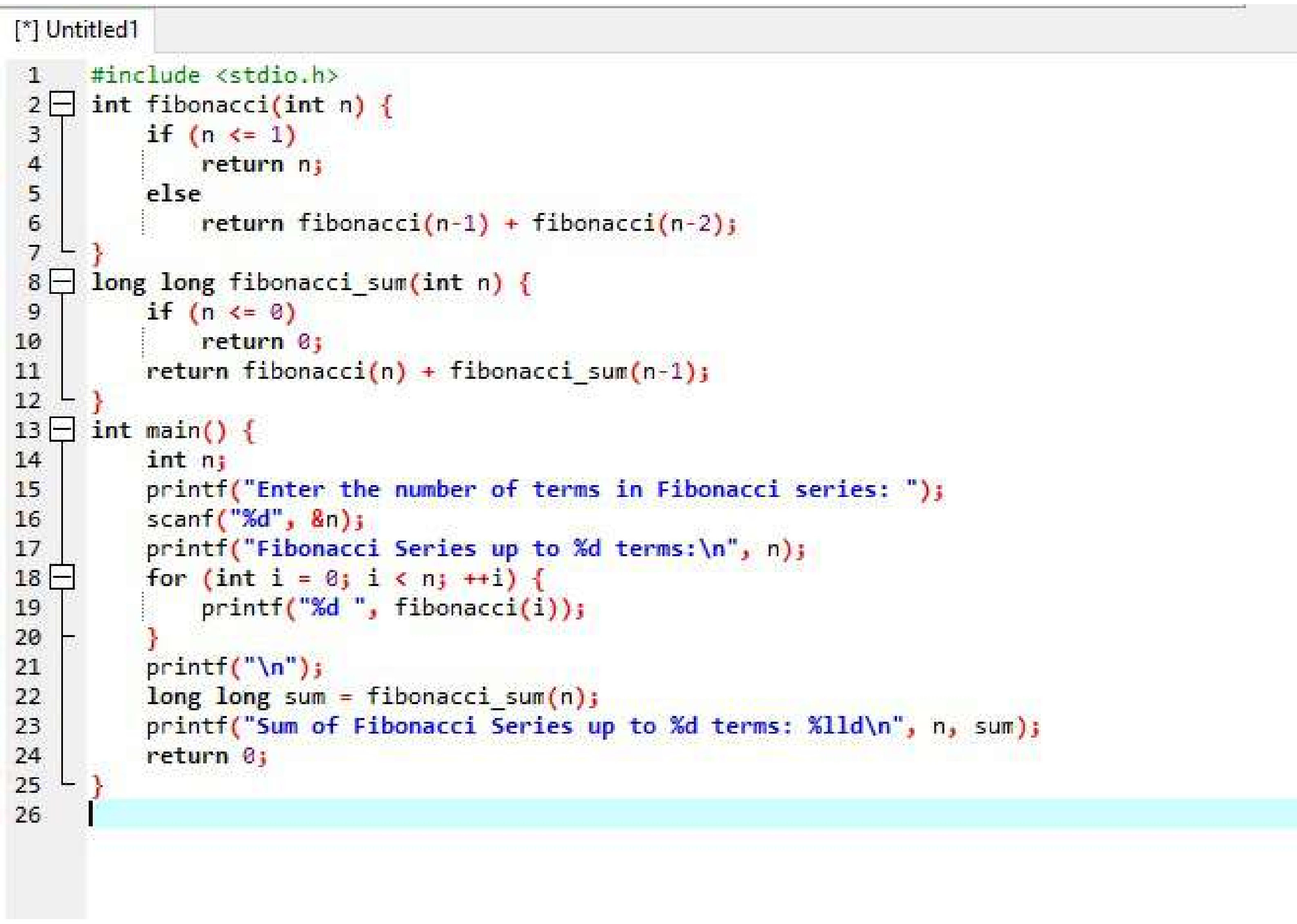


[*] Untitled1

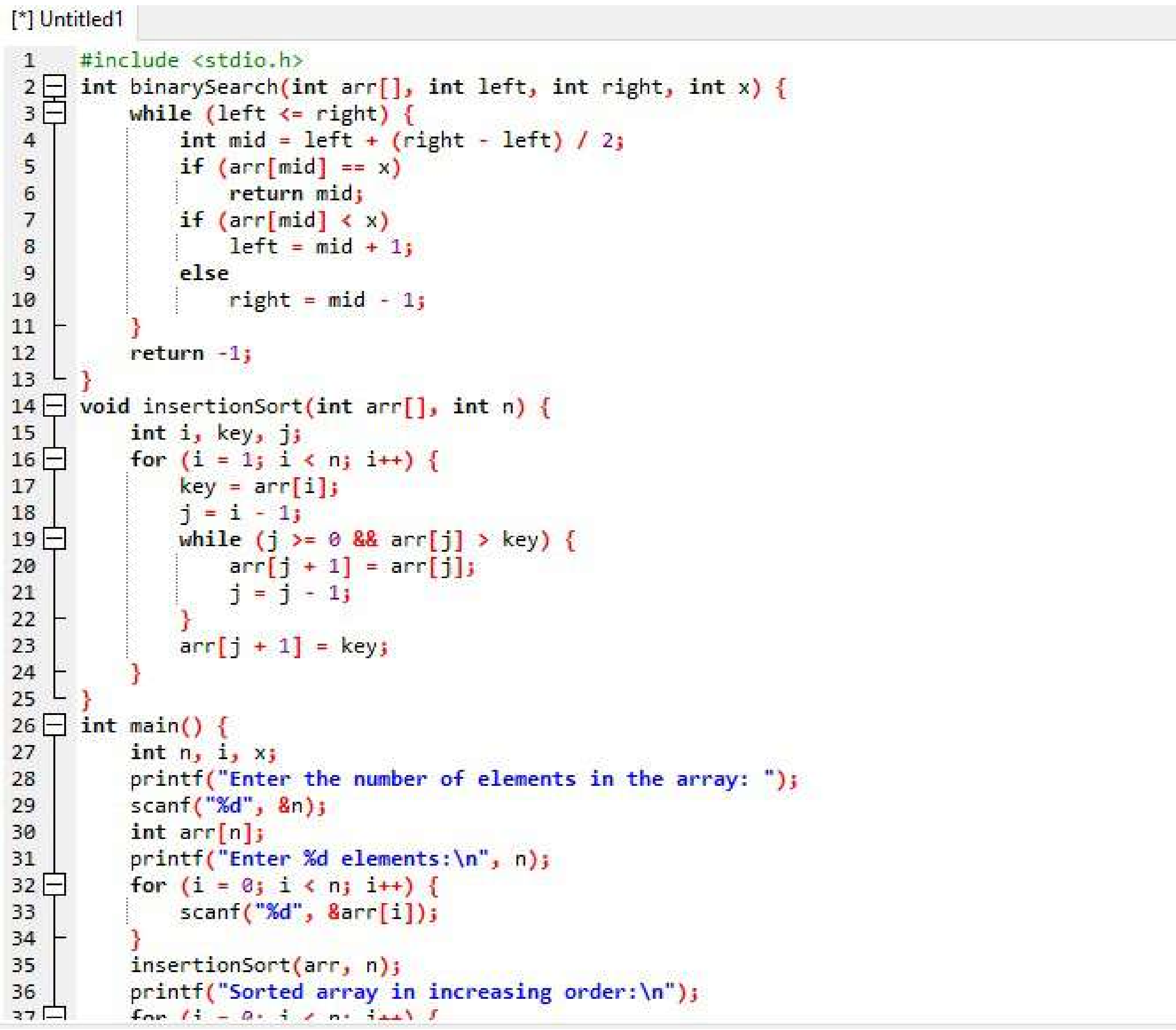
```c
#include <stdio.h>
int main()
{
    int arr[] = {1, 2, 3, 4, 2, 7, 8, 8, 3};
    int length = sizeof(arr)/sizeof(arr[0]);
    printf("Duplicate elements in given array: \n");
    for(int i = 0; i < length; i++) {
        for(int j = i + 1; j < length; j++) {
            if(arr[i] == arr[j])
                printf("%d\n", arr[j]);
        }
    }
    return 0;
}
```



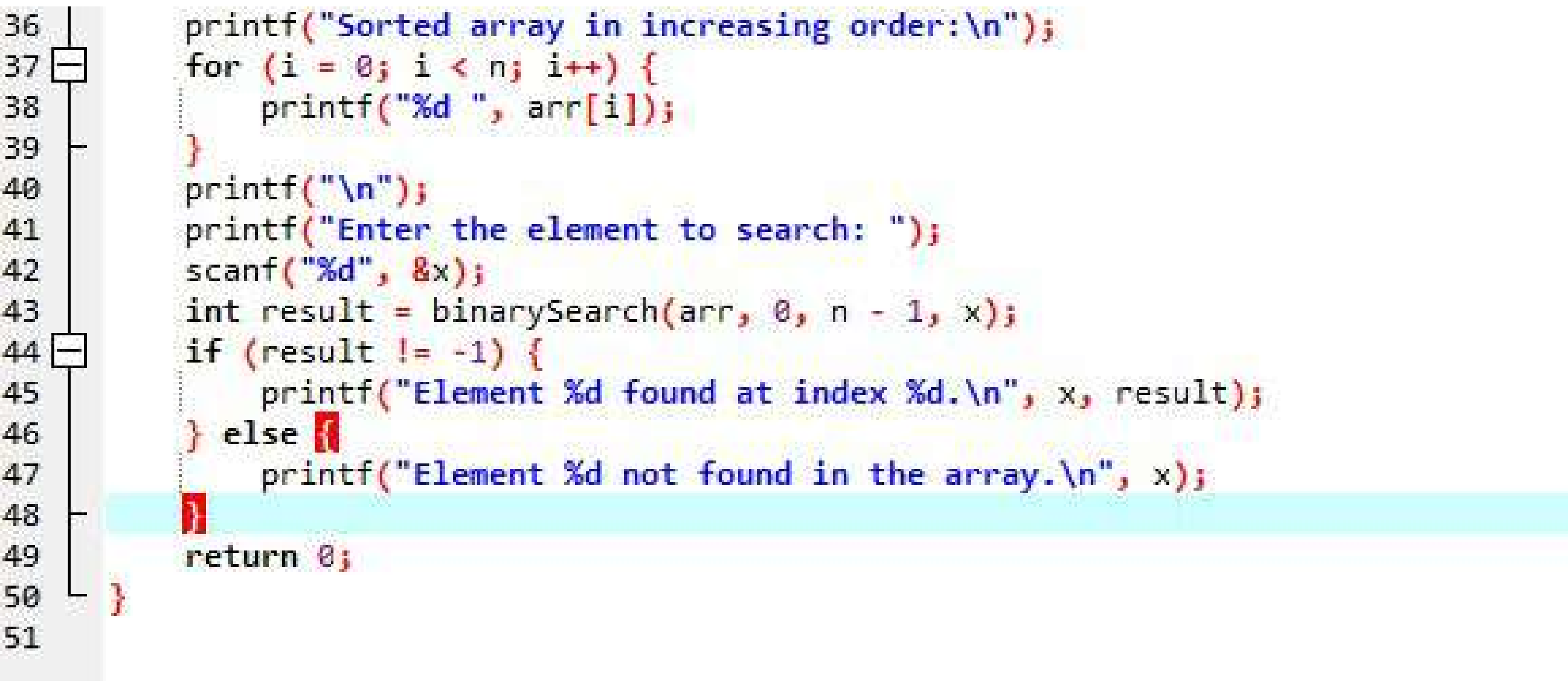
```c
#include <limits.h>
#include <stdio.h>
void findMinimumMaximum(int arr[], int N)
{
    int i;
    int minE = INT_MAX, maxE = INT_MIN;
    for (i = 0; i < N; i++) {
    if (arr[i] < minE) {
            minE = arr[i];
        }
        if (arr[i] > maxE) {
            maxE = arr[i];
        }
    }
    printf("The minimum element is %d", minE);
    printf("\n");
    printf("The maximum element is %d", maxE);

    return;
}
int main()
{
    int arr[] = { 1, 2, 4, -1 };
    int N = sizeof(arr) / sizeof(arr[0]);
    findMinimumMaximum(arr, N);
    return 0;
}
```


[*] Untitled1

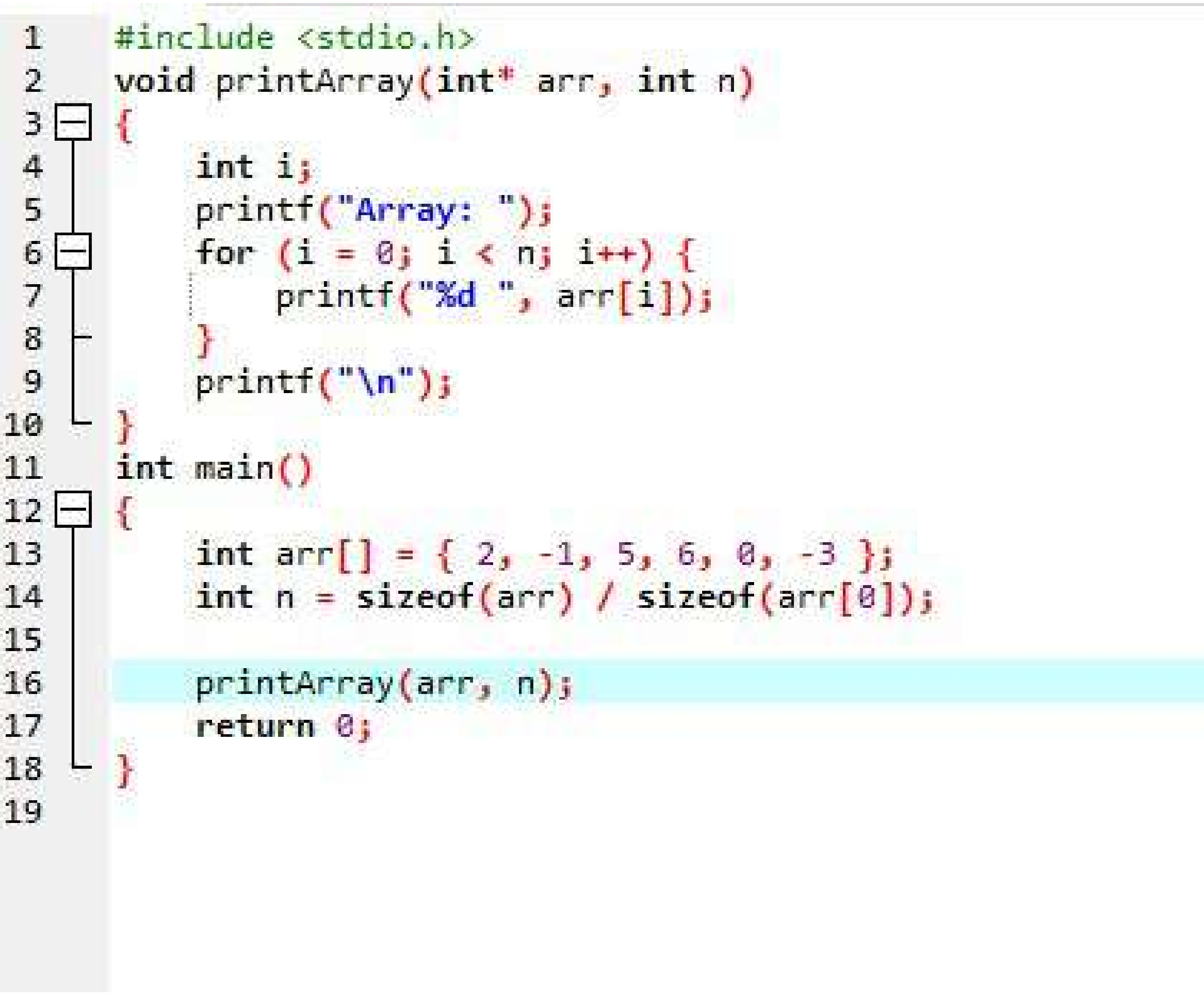
```c
#include <stdio.h>
int fibonacci(int n) {
    if (n <= 1)
        return n;
    else
        return fibonacci(n-1) + fibonacci(n-2);
}
long long fibonacci_sum(int n) {
    if (n <= 0)
        return 0;
    return fibonacci(n) + fibonacci_sum(n-1);
}
int main() {
    int n;
    printf("Enter the number of terms in Fibonacci series: ");
    scanf("%d", &n);
    printf("Fibonacci Series up to %d terms:\n", n);
    for (int i = 0; i < n; ++i) {
        printf("%d ", fibonacci(i));
    }
    printf("\n");
    long long sum = fibonacci_sum(n);
    printf("Sum of Fibonacci Series up to %d terms: %lld\n", n, sum);
    return 0;
}

```



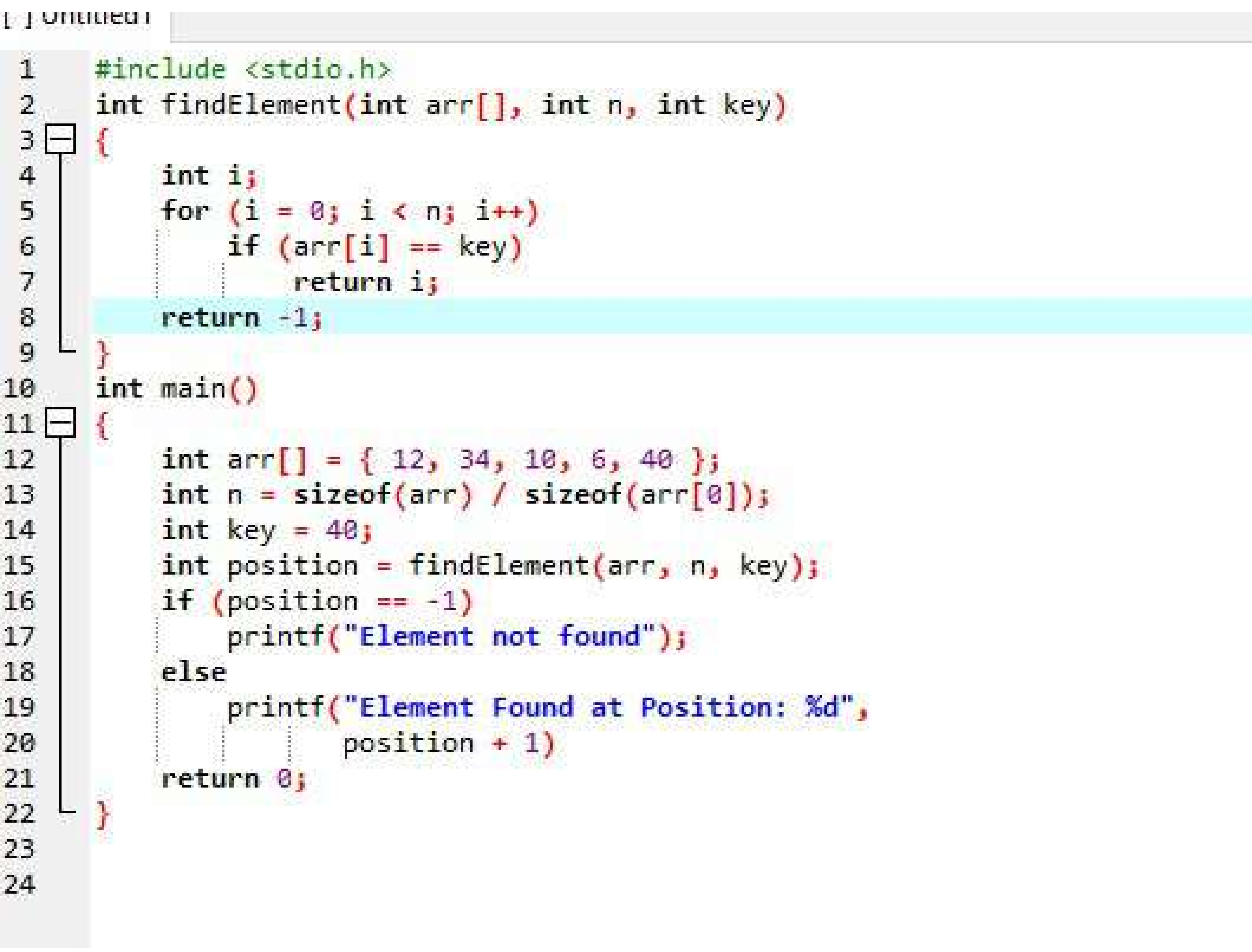
```c
#include <stdio.h>
int binarySearch(int arr[], int left, int right, int x) {
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (arr[mid] == x)
            return mid;
        if (arr[mid] < x)
            left = mid + 1;
        else
            right = mid - 1;
    }
    return -1;
}
void insertionSort(int arr[], int n) {
    int i, key, j;
    for (i = 1; i < n; i++) {
        key = arr[i];
        j = i - 1;
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}
int main() {
    int n, i, x;
    printf("Enter the number of elements in the array: ");
    scanf("%d", &n);
    int arr[n];
    printf("Enter %d elements:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    insertionSort(arr, n);
    printf("Sorted array in increasing order:\n");
    for (i = 0; i < n; i++) {
```



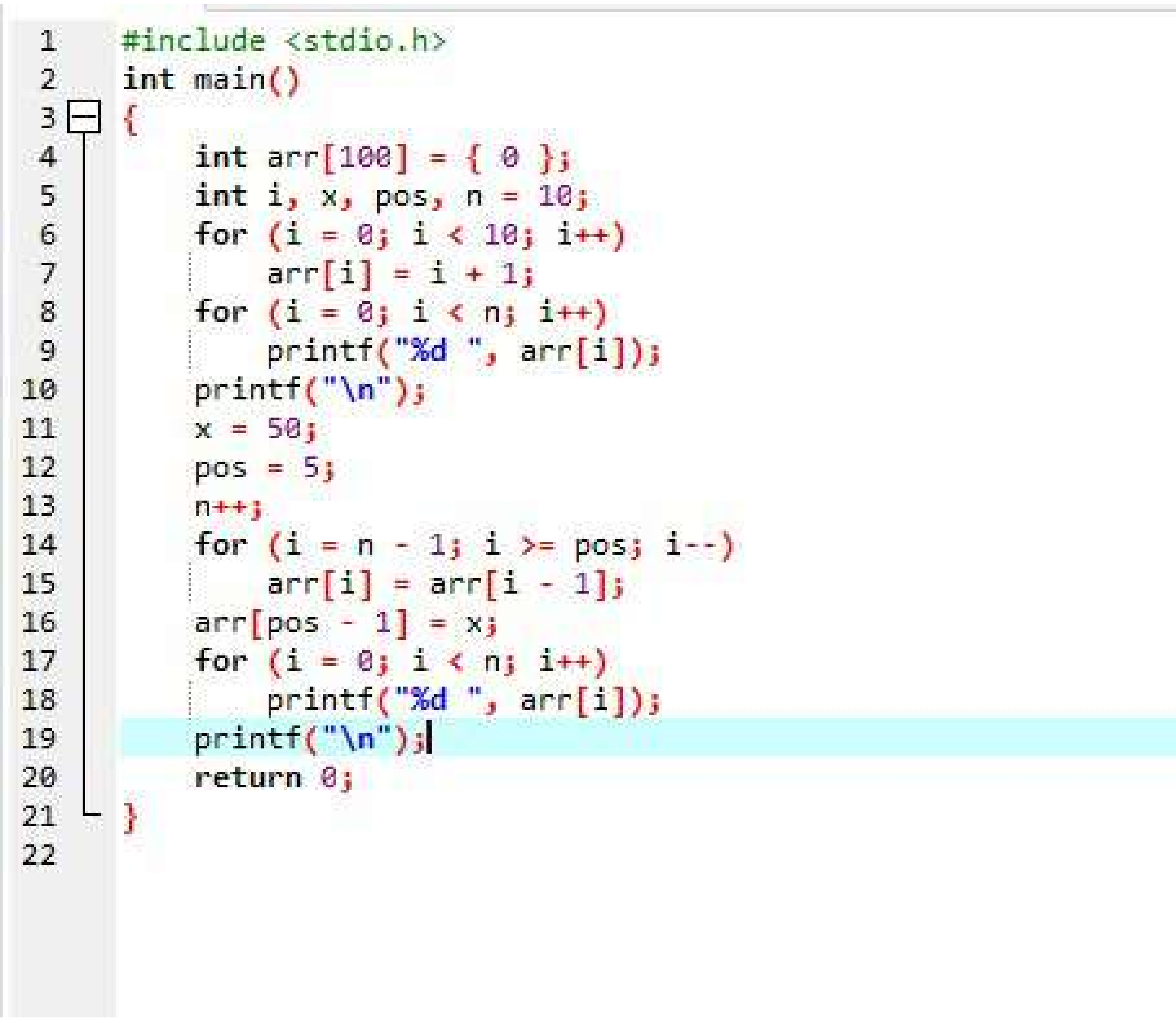
```c
    printf("Sorted array in increasing order:\n");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
    printf("Enter the element to search: ");
    scanf("%d", &x);
    int result = binarySearch(arr, 0, n - 1, x);
    if (result != -1) {
        printf("Element %d found at index %d.\n", x, result);
    } else {
        printf("Element %d not found in the array.\n", x);
    }
    return 0;
}

```



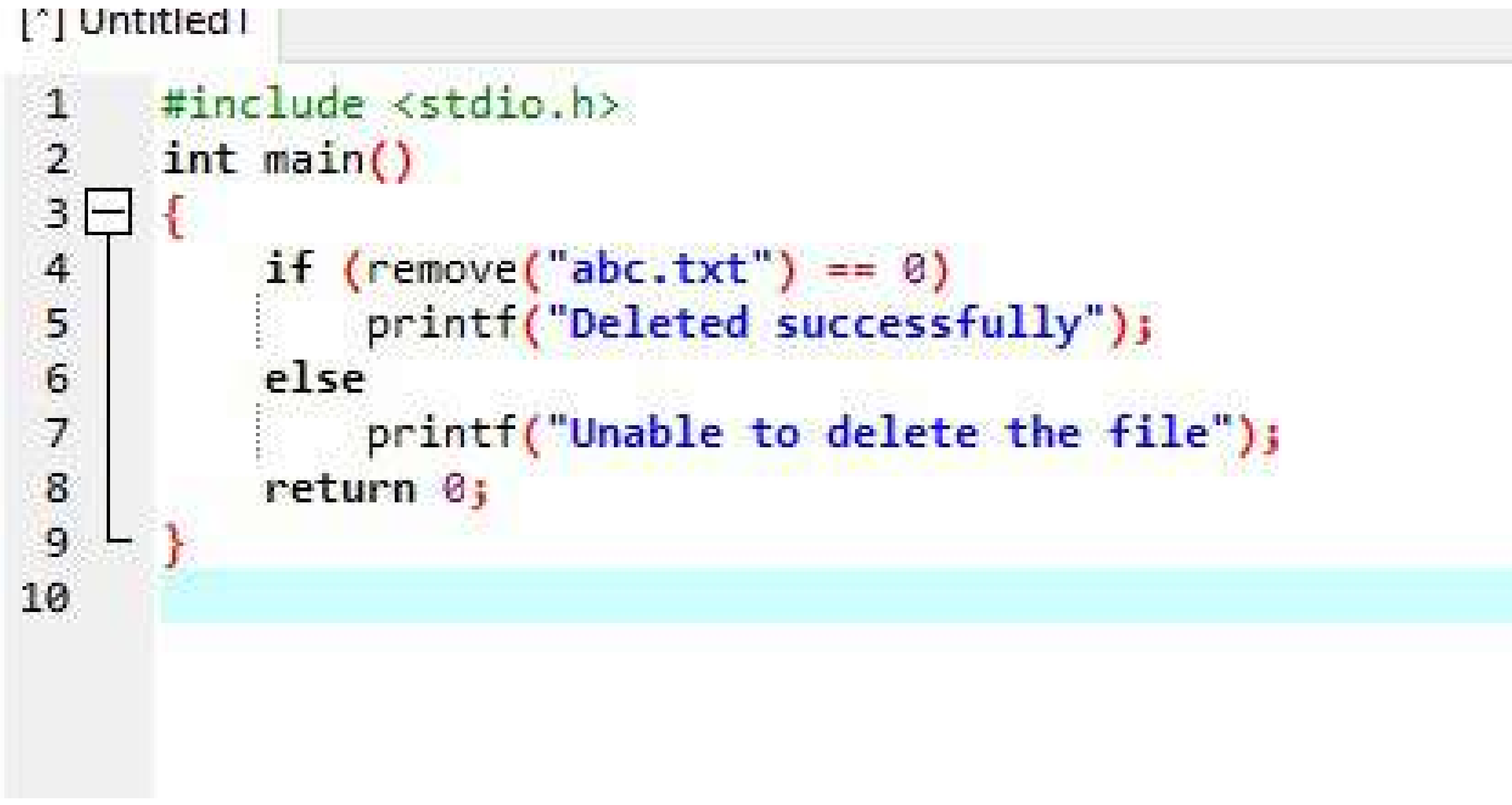
```c
#include <stdio.h>
void printArray(int* arr, int n)
{
    int i;
    printf("Array: ");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
}
int main()
{
    int arr[] = { 2, -1, 5, 6, 0, -3 };
    int n = sizeof(arr) / sizeof(arr[0]);

    printArray(arr, n);
    return 0;
}
```



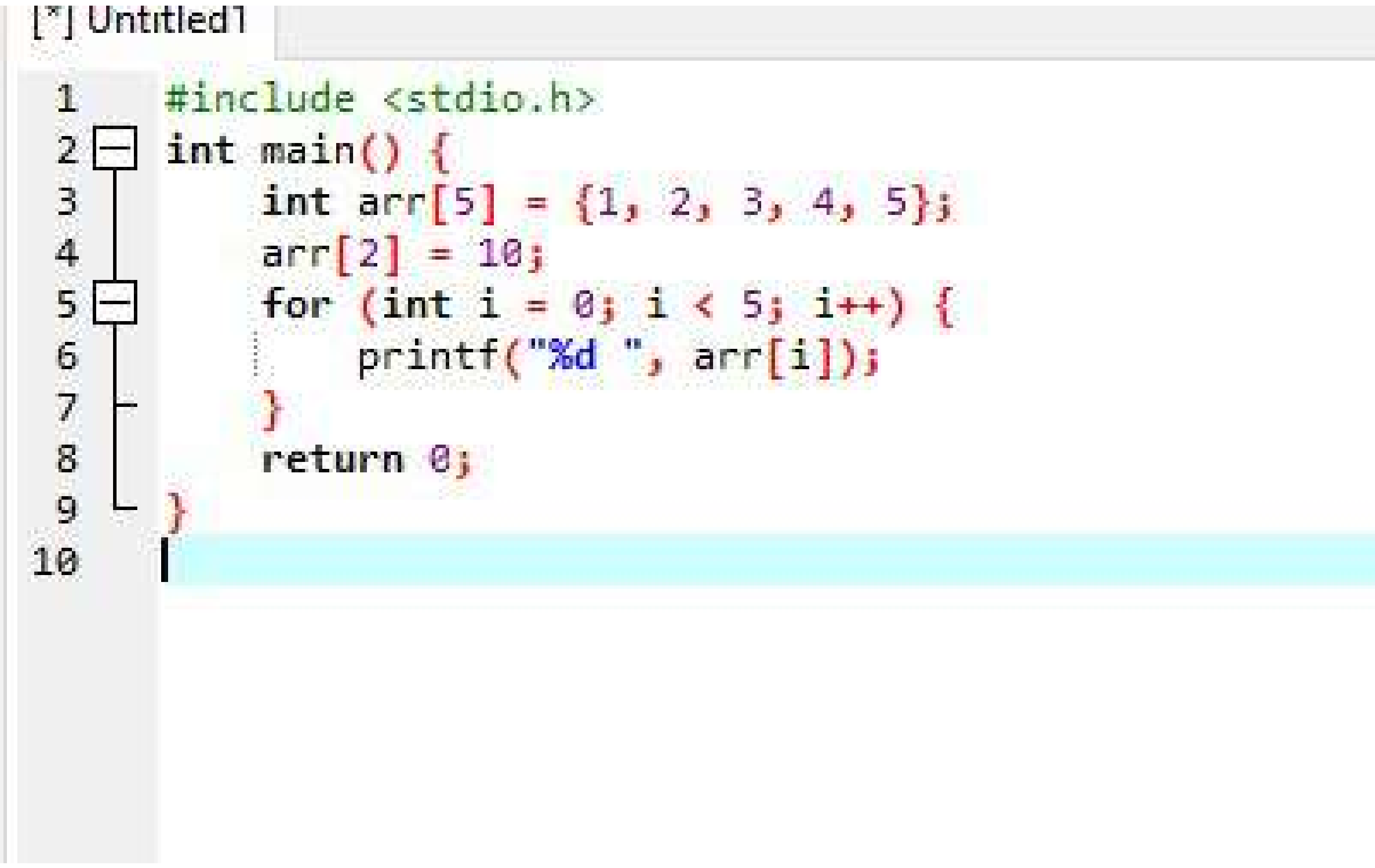
```
#include <stdio.h>
int findElement(int arr[], int n, int key)
{
    int i;
    for (i = 0; i < n; i++)
        if (arr[i] == key)
            return i;
    return -1;
}
int main()
{
    int arr[] = { 12, 34, 10, 6, 40 };
    int n = sizeof(arr) / sizeof(arr[0]);
    int key = 40;
    int position = findElement(arr, n, key);
    if (position == -1)
        printf("Element not found");
    else
        printf("Element Found at Position: %d",
                position + 1)
    return 0;
}
```



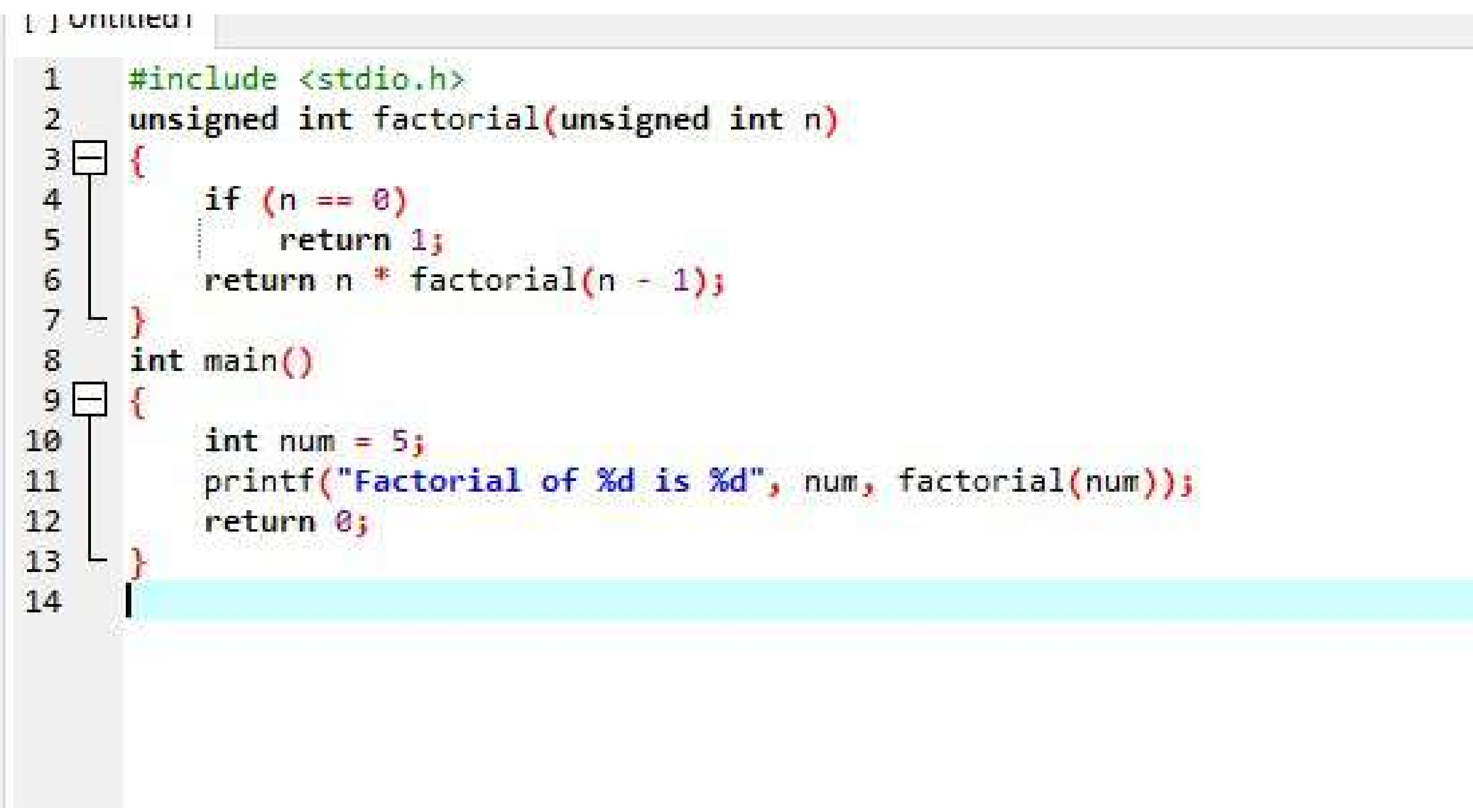
```c
#include <stdio.h>
int main()
{
    int arr[100] = { 0 };
    int i, x, pos, n = 10;
    for (i = 0; i < 10; i++)
        arr[i] = i + 1;
    for (i = 0; i < n; i++)
        printf("%d ", arr[i]);
    printf("\n");
    x = 50;
    pos = 5;
    n++;
    for (i = n - 1; i >= pos; i--)
        arr[i] = arr[i - 1];
    arr[pos - 1] = x;
    for (i = 0; i < n; i++)
        printf("%d ", arr[i]);
    printf("\n");
    return 0;
}
```



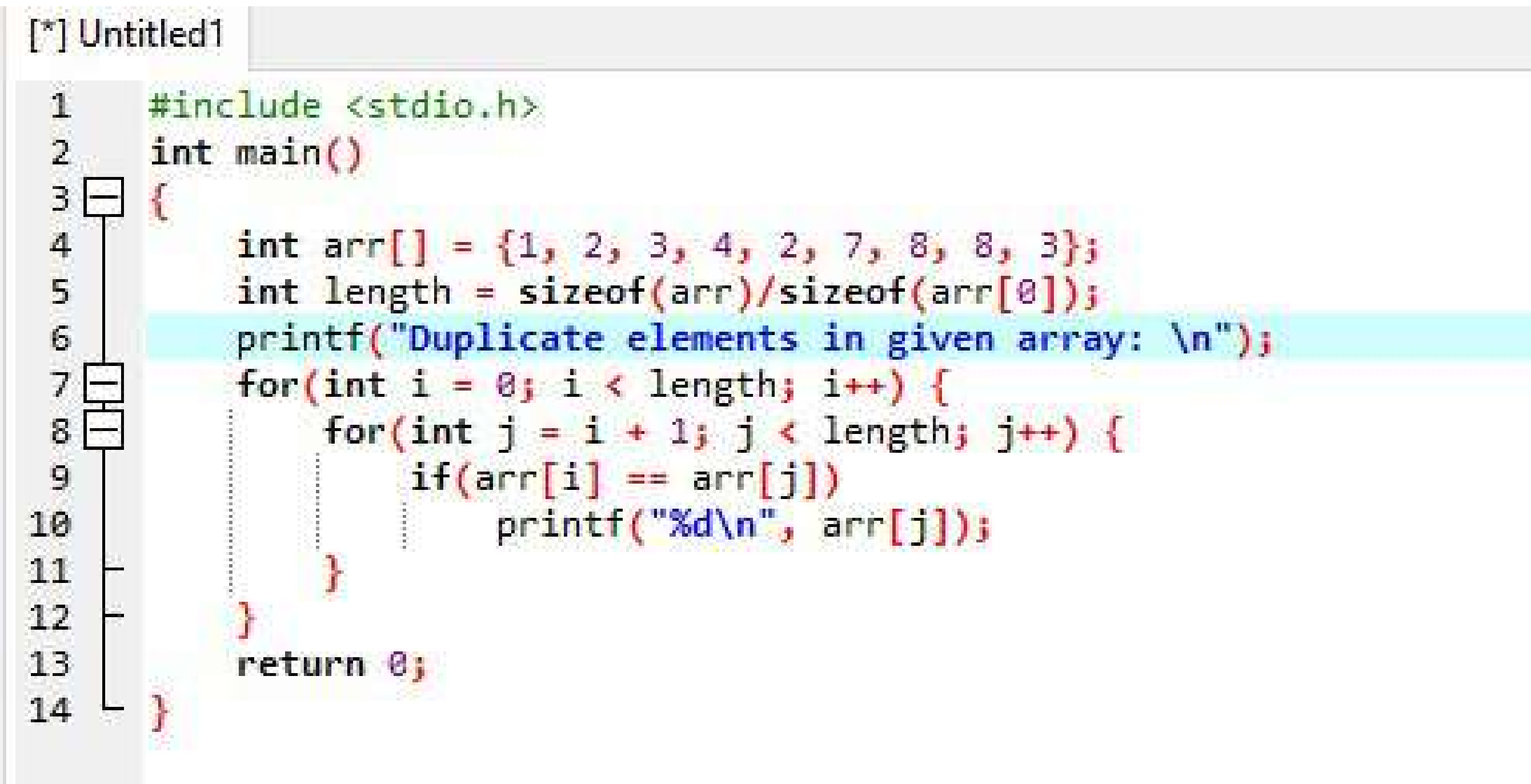
```c
#include <stdio.h>
int main()
{
	if (remove("abc.txt") == 0)
		printf("Deleted successfully");
	else
		printf("Unable to delete the file");
	return 0;
}
```



```c
#include <stdio.h>
int main() {
    int arr[5] = {1, 2, 3, 4, 5};
    arr[2] = 10;
    for (int i = 0; i < 5; i++) {
        printf("%d ", arr[i]);
    }
    return 0;
}
```

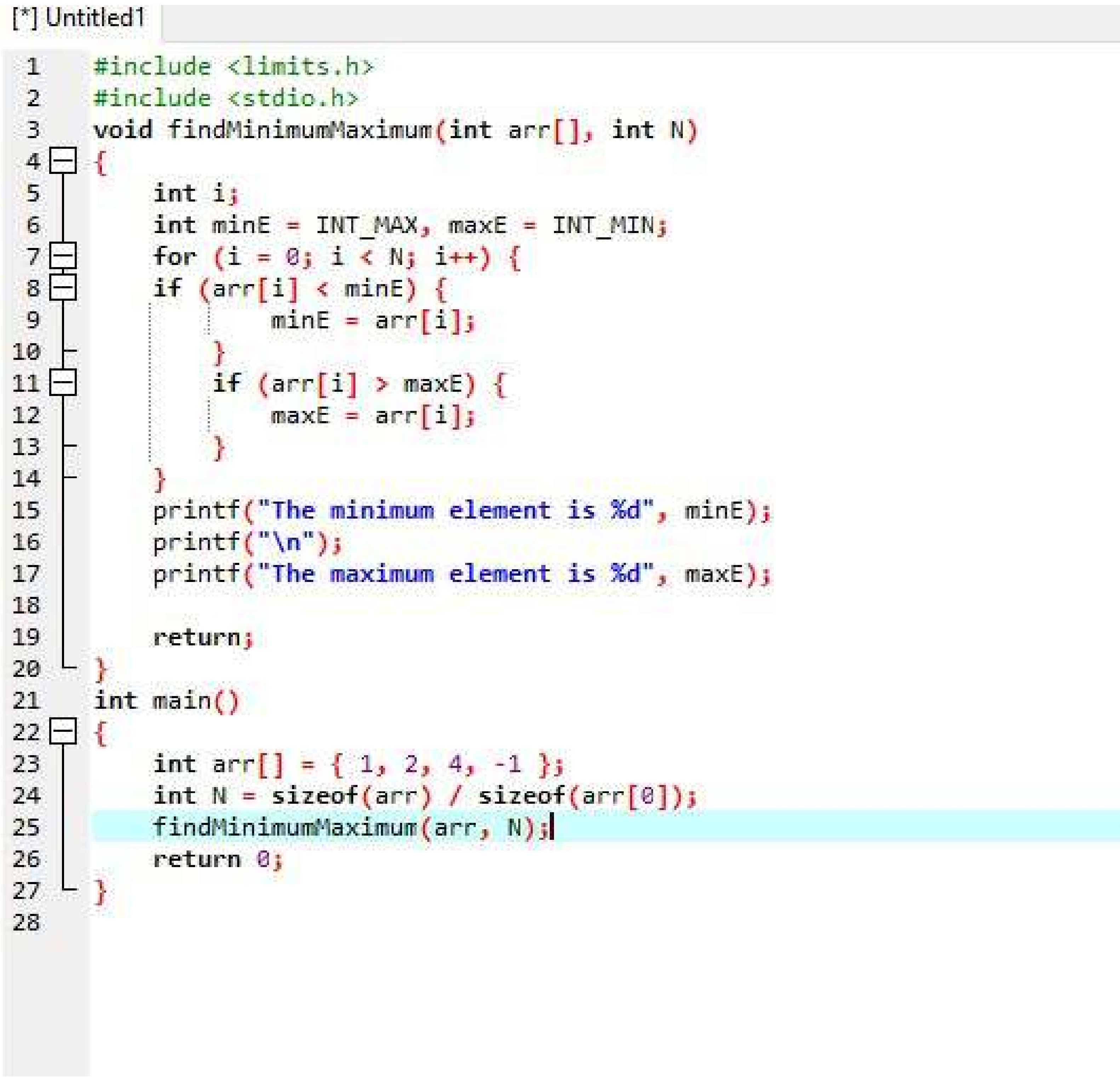


```c
#include <stdio.h>
unsigned int factorial(unsigned int n)
{
    if (n == 0)
        return 1;
    return n * factorial(n - 1);
}
int main()
{
    int num = 5;
    printf("Factorial of %d is %d", num, factorial(num));
    return 0;
}
```

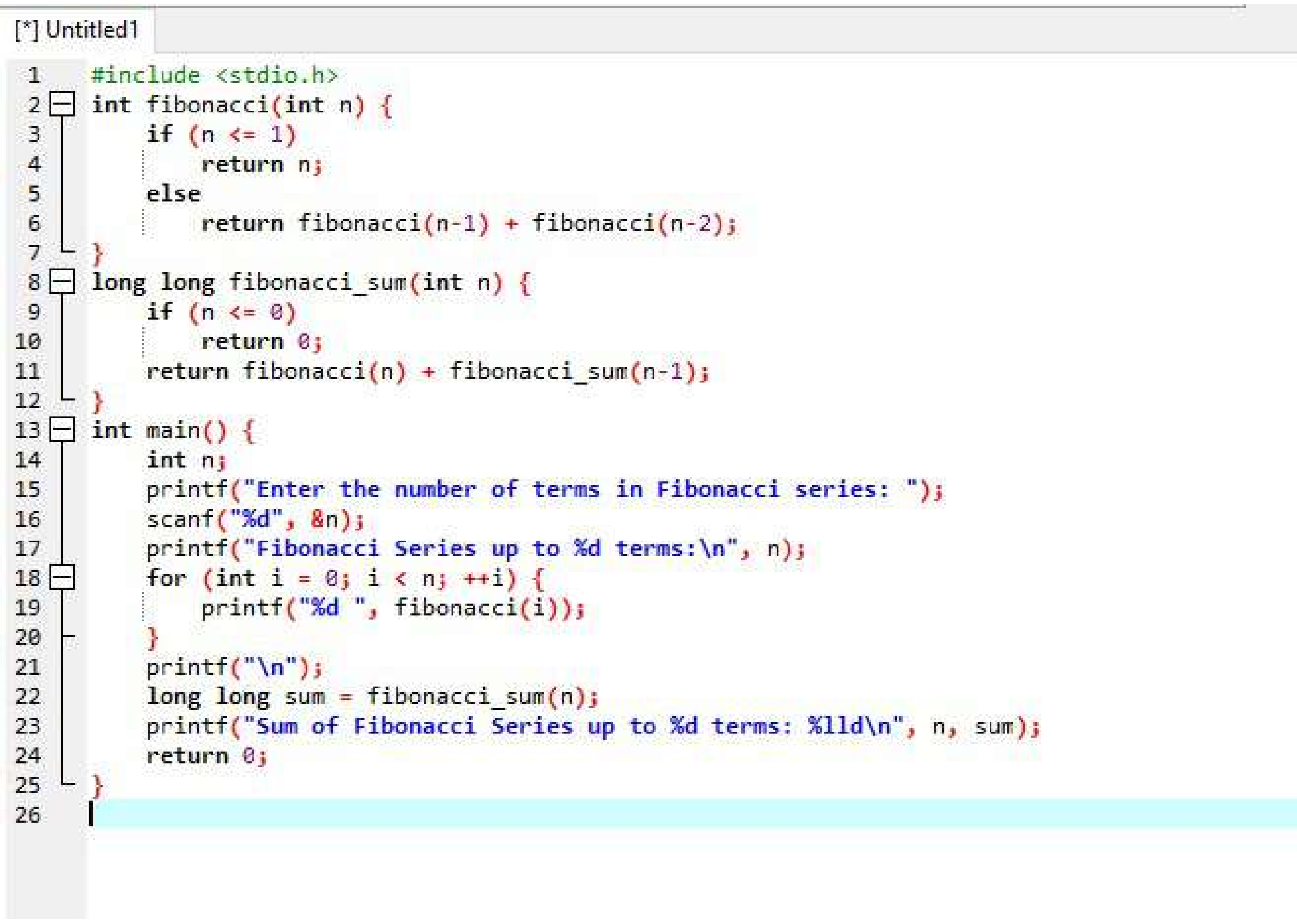


[*] Untitled1

```c
#include <stdio.h>
int main()
{
    int arr[] = {1, 2, 3, 4, 2, 7, 8, 8, 3};
    int length = sizeof(arr)/sizeof(arr[0]);
    printf("Duplicate elements in given array: \n");
    for(int i = 0; i < length; i++) {
        for(int j = i + 1; j < length; j++) {
            if(arr[i] == arr[j])
                printf("%d\n", arr[j]);
        }
    }
    return 0;
}
```

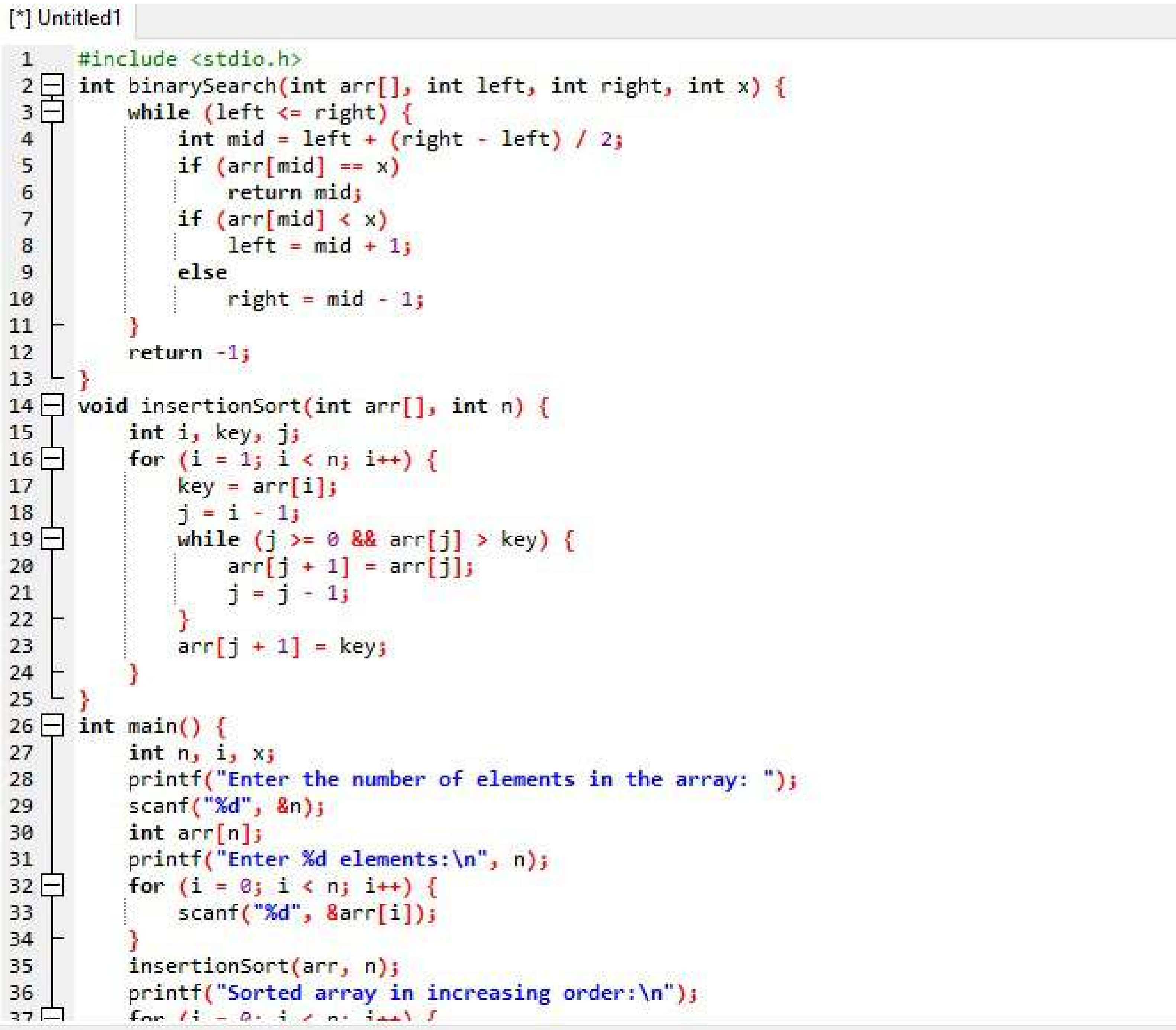


```c
#include <limits.h>
#include <stdio.h>
void findMinimumMaximum(int arr[], int N)
{
    int i;
    int minE = INT_MAX, maxE = INT_MIN;
    for (i = 0; i < N; i++) {
    if (arr[i] < minE) {
            minE = arr[i];
        }
        if (arr[i] > maxE) {
            maxE = arr[i];
        }
    }
    printf("The minimum element is %d", minE);
    printf("\n");
    printf("The maximum element is %d", maxE);

    return;
}
int main()
{
    int arr[] = { 1, 2, 4, -1 };
    int N = sizeof(arr) / sizeof(arr[0]);
    findMinimumMaximum(arr, N);
    return 0;
}
```



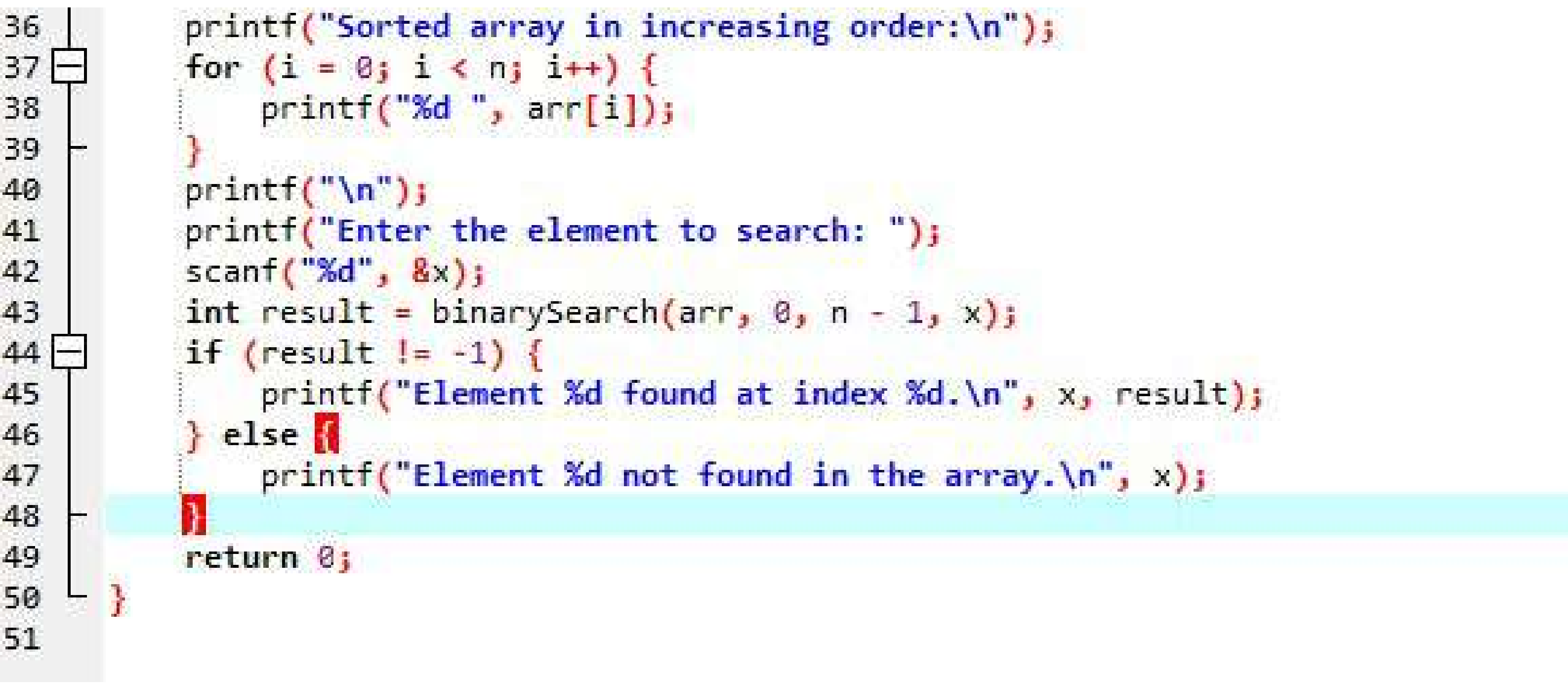
[*] Untitled1

```c
#include <stdio.h>
int fibonacci(int n) {
    if (n <= 1)
        return n;
    else
        return fibonacci(n-1) + fibonacci(n-2);
}
long long fibonacci_sum(int n) {
    if (n <= 0)
        return 0;
    return fibonacci(n) + fibonacci_sum(n-1);
}
int main() {
    int n;
    printf("Enter the number of terms in Fibonacci series: ");
    scanf("%d", &n);
    printf("Fibonacci Series up to %d terms:\n", n);
    for (int i = 0; i < n; ++i) {
        printf("%d ", fibonacci(i));
    }
    printf("\n");
    long long sum = fibonacci_sum(n);
    printf("Sum of Fibonacci Series up to %d terms: %lld\n", n, sum);
    return 0;
}
```

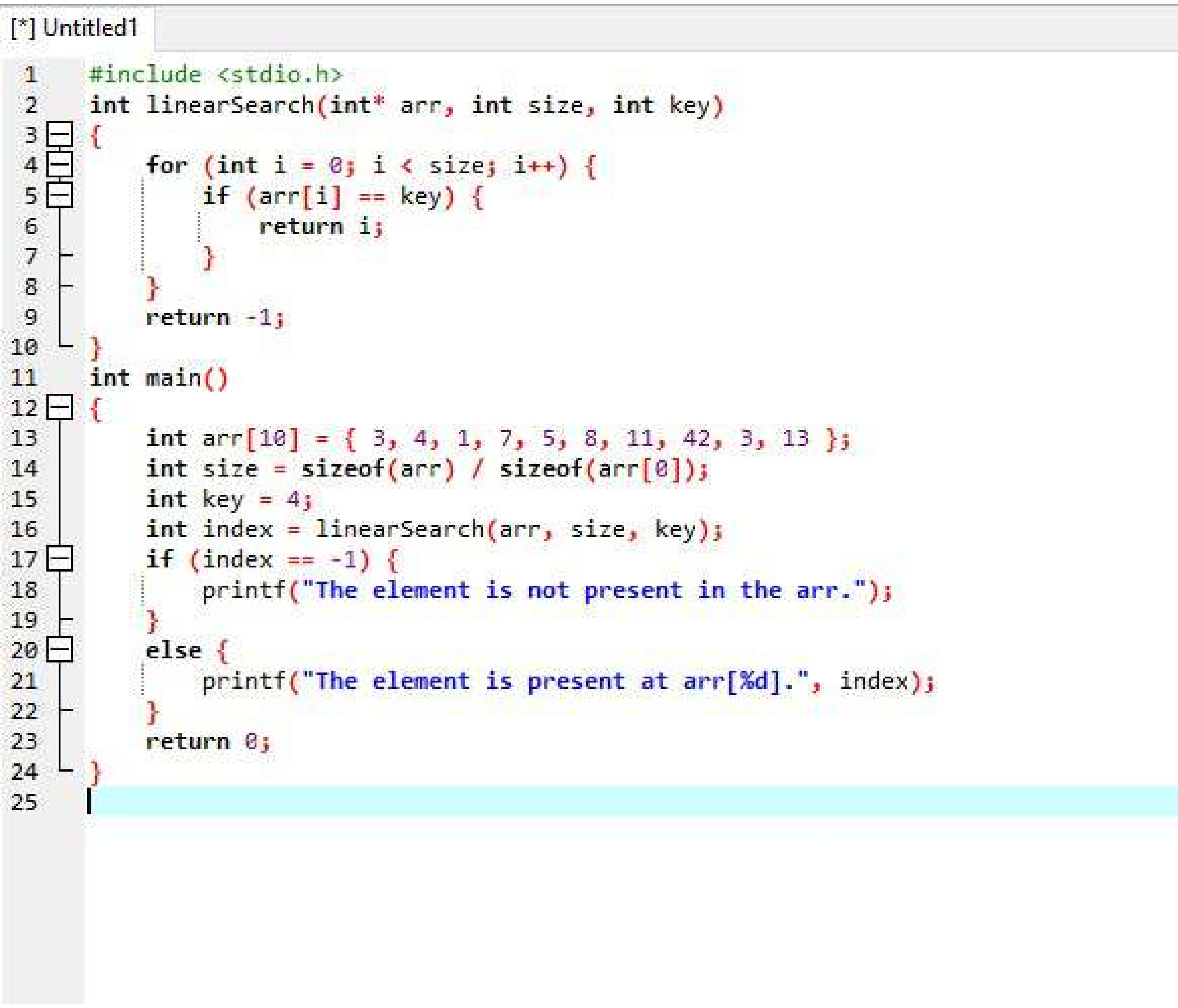


[*] Untitled1

```c
#include <stdio.h>
int binarySearch(int arr[], int left, int right, int x) {
    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (arr[mid] == x)
            return mid;
        if (arr[mid] < x)
            left = mid + 1;
        else
            right = mid - 1;
    }
    return -1;
}
void insertionSort(int arr[], int n) {
    int i, key, j;
    for (i = 1; i < n; i++) {
        key = arr[i];
        j = i - 1;
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}
int main() {
    int n, i, x;
    printf("Enter the number of elements in the array: ");
    scanf("%d", &n);
    int arr[n];
    printf("Enter %d elements:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
    insertionSort(arr, n);
    printf("Sorted array in increasing order:\n");
    for (i = 0; i < n; i++) {
```



```c
    printf("Sorted array in increasing order:\n");
    for (i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
    printf("Enter the element to search: ");
    scanf("%d", &x);
    int result = binarySearch(arr, 0, n - 1, x);
    if (result != -1) {
        printf("Element %d found at index %d.\n", x, result);
    } else {
        printf("Element %d not found in the array.\n", x);
    }
    return 0;
}
```


[*] Untitled1

```c
#include <stdio.h>
int linearSearch(int* arr, int size, int key)
{
    for (int i = 0; i < size; i++) {
        if (arr[i] == key) {
            return i;
        }
    }
    return -1;
}
int main()
{
    int arr[10] = { 3, 4, 1, 7, 5, 8, 11, 42, 3, 13 };
    int size = sizeof(arr) / sizeof(arr[0]);
    int key = 4;
    int index = linearSearch(arr, size, key);
    if (index == -1) {
        printf("The element is not present in the arr.");
    }
    else {
        printf("The element is present at arr[%d].", index);
    }
    return 0;
}

```