**TASK 1:**

#include<iostream>

using namespace std;

int main() {

int size;

int i = 0;

cout << "input size of the array:";

cin >> size;

int\* array = new int[size]; //dynamic memory allocation creating a dynamic array

while (true) { //using while loop to keep the statement running

try { //exception handling

if (i < size) { //if i<size then we will keep on taking input in array

cout << "input in array:";

cin >> array[i];

i++;

}

if (i<0 || i>size - 1) { //else throw exception

throw(i);

}

}

catch (int value) {

cout << "index value increase from the given size. It is out of bound" << endl; //catch statement. print index out of bound if size gets greater

break;

}

}

for (int i = 0; i < size; i++) {

cout << array[i] << " "; //printing the array

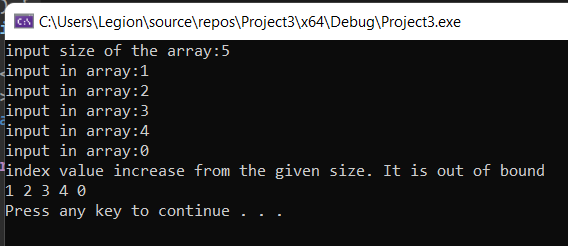
}

cout << endl;

system("pause");

return 0;

}

****

**TASK 2:**

**Part A:**

variable “number” is not initialized with any value. You must initialize it with any value then the issue is resolved. It then prints “number” and the program is executed.

Solution:

#include<iostream>

using namespace std;

int main()

{

**int\* number=0; //solution**

cout << number << endl;

}

**Part B:**

variables or pointers of different data types cannot be made equal i.e. Long and double cannot be made equal due to different data types. Making the data type same resolves the issue. Also both are uninitialized.

Solution:

#include<iostream>

using namespace std;

int main()

{

long\* realPtr=0;

long\* integerPtr=0;

integerPtr = realPtr;

}

**Part C:**

a value of type "int" cannot be assigned to an entity of type "int \*".

Solution :

#include<iostream>

using namespace std;

int main()

{

int\* x, y;

\*x = y;

}

**Part D:**

Solution :

#include <iostream>

using namespace std;

int main()

{

char s[] = "this is a character array";

for(int i=0;s[i]!='\0';i++)

cout << s[i] <<"";

}

**Part F:**

a value of type "double \*" cannot be used to initialize an entity of type "double". 'initializing': cannot convert from 'double \*' to 'double'

**TASK 3:**

**PART A:**

The code creates two char array of size 80. String1 and string2. Both of size 80. The function mystery1 creates two char pointer that each point at string1 and string2 respectively. After the calling of function string1 now has string1 and string2 both values stored in them. Then it is printed on the screen.

#include <iostream>

using namespace std;

void mystery1(char\*, const char\*); // prototype

int main()

{

char string1[80]; // The code creates two char array of size 80. String1 and string2. Both of size 80

char string2[80];

cout << "Enter two strings: ";

cin >> string1 >> string2; //taking input

mystery1(string1, string2);

cout << string1 << endl; // Then it is printed on the screen.

} // end main// What does this function do?

void mystery1(char\* s1, const char\* s2) // The function mystery1 creates two char pointer that each point at string1 and string2 respectively

{

while (\*s1 != '\0')

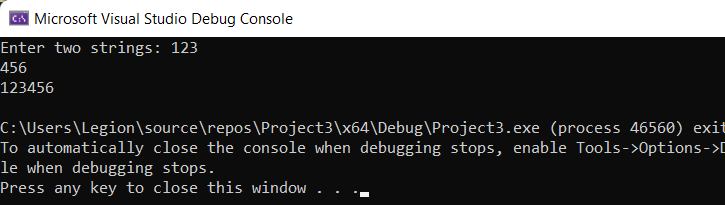
++s1;

for (; \*s1 = \*s2; ++s1, ++s2)

;// empty statement

} // end function mystery1

//. After the calling of function string1 now has string1 and string2 both values stored in them.



**PART B:**

#include <iostream>

using namespace std;

int mystery2(const char\*); // prototype

int main()

{

char string1[80];

cout << "Enter a string: ";

cin >> string1;

cout << mystery2(string1) << endl; //printing the return value x from the mystery2 function.

} // end main

// What does this function do?

int mystery2(const char\* s)

{

int x;

for (x = 0; \*s != '\0'; ++s)

++x;

return x;

} // end function mystery2

//What the function does is that it counts the number of characters in the char array name string1 of size 80 “string1[80]”. But note that it does not count the number of characters after space “ ”. For example if put “abcd” in string1 after calling the function it will return “4”. However if I put “abcd abcd” in string1 it will still return “4” as the logic in the mystery2 function does not count the entries in the char array after space “ ”

**Text

Description automatically generated**

**Text

Description automatically generated**