**Task 1**

#include <iostream>

using namespace std;

int main() {

    cout << "Size of char: " << sizeof(char) << " byte" << endl;

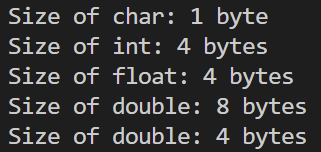
    cout << "Size of int: " << sizeof(int) << " bytes" << endl;

    cout << "Size of float: " << sizeof(float) << " bytes" << endl;

    cout << "Size of double: " << sizeof(double) << " bytes" << endl;

    cout << "Size of double: " << sizeof(long) << " bytes" << endl;

}



**Task 2**

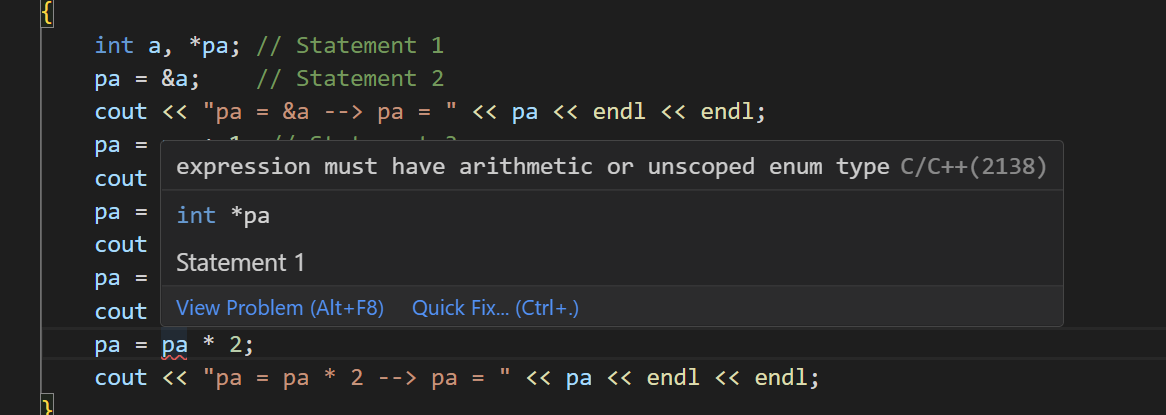
1. **Why does the memory address stored in pointer “pa” vary by 4?**

That is because the memory occupied by an int variable is 4 bytes

1. **Will the address still vary by 4 if the data type of the above mentioned code changed from “int” to “long”? Explain your answer.**

In that case, the address will vary by 4 bytes as well as long data type also occupies 4 bytes

1. **If we try to multiply the address pointed to by “pa” what will happen? Is this logically or programmatically correct? Attach screen shot of the output you get when you try this multiplication.**

****

We get an error as multiplication is not allowed on pointers in cpp

**Task 3**

#include <iostream>

using namespace std;

int main()

{

    int list[5]={3,6,9,12,15};

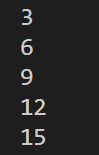
    int \*pArr= list;

    for (int i; i< 5; i++){

        cout<< \*(pArr + i) << endl;

    };

}



**Task 4**

1. 1

4

7

9

1. 4 4 5 7

**Task 5**

#include <iostream>

using namespace std;

void swap1(int \*a, int \*b){

    int temp = \*a;

    \*a = \*b;

    \*b = temp;

};

void swap2(int \*\*a, int \*\*b){

    int temp = \*\*a;

    \*\*a = \*\*b;

    \*\*b = temp;

};

int main()

{

    int a=5, b=10;

    int \*pa=&a; //pa and pb are pointer variables of type int.

    int \*pb=&b;

    swap1(pa, pb);

    cout<<"a:  "<< a << "  b: "<<b<<endl;

    int \*\*ppa=&pa; //ppa and ppb are called double pointers or pointers-to-pointers.

    int \*\*ppb=&pb;

    swap2(ppa, ppb);

    cout<<"a:  "<< a << "  b: "<<b<<endl;

}

