CSE 232 Sample Questions

Michigan State University

February 9, 2024

How to use this document

This document will be able to give you sample questions to understand the concepts in more detail. Try to look at the code, try to answer what the problem will output. Copy the code into $test\_set.cpp$ file and then run the below commands.

References and pointers will be used to pass the data without copying the total data.

```
$ g++ —std=c++20 test_set.cpp —o test_ans.o
$ ./test_ans.o
```

References & Pointers

What are the common themes between references and pointers. To understand how the pointers and reference work it is important to understand the memory and how they are stored.

Listing 1: Use case of reference & pointers

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

using namespace std;


int main(){
    int x =231;
    int& r = x;
    int *ptr = &x;
    cout  << &x << " == " << &r << " == " << ptr << endl; // all these are the same value to access memory
    location of x
    /*
        value of x can be changed in multiple ways.
    */
    x = 232;
    *ptr = 232;
    r = 232;
    /*
        above 3 lines are equivalent
    */

   //one difference include the following.

    r = NULL; // NOT advaisable to do it. you can try this, it will give rise to warining but here you are
    assining a value to x not to r, r is just a new symbol for x
    /*
        explicit conversion takes place NULL is converted to the type of x, which is int so Null is converted to 0
    */
    cout << x <<  endl; // will print 0

    ptr = NULL; // this is ok, no warning and legal to do it
    cout << *ptr <<  endl; // However, but you can't dereference it, if you try to do so you will get a
    segmentation fault
    /*
        if you are seing a segmentation faut run time error, it means that you are trying to access a
        memory location that you are not allowed to access (more common when you are using lists)
    */
    return 0;
}
```

What will happen when you pass a arguments to a function.

```cpp
#include <stdio.h>
#include <iostream>
using namespace std;

void some_function_name(string func_variable){
    cout << &func_variable; // check the address of func_variable Is it equal to the main variable
    /*
        The answer is No, func variables copies all the content from main_variable into a new memory location
    given above.
    */
}
void some_function_name_reference(string &func_variable){
    cout << &func_variable; // check the address of func_variable Is the equal to the main variable
    /*
        Here func_variable is a new name to main_variable so the address will be the same.
    */
}

void some_function_name_pointer(string *func_ptr){
    cout << func_ptr;
    /*
        Here func_ptr is the one that has passed on from the main function, so func_ptr points to the
    main_variable memory location.
    */
    cout << &func_ptr;
    /*
        However the address of func_ptr and main_ptr are different so what happend here, func_ptr copies all the
    data from main_ptr. They are on different memory locations but contains same value, address of main_variable
    */
}

int main(){
    string main_variable = "main";
    string main_ptr = &main_variable
    some_function_name(main_variable);
    some_function_name_reference(main_variable);
    some_function_name_pointer(main_ptr);
}
```

Listing 2: Use case of reference & pointers

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

using namespace std;


void pass_by_value(string v){
    v = "modified value pass_by_value";
}

void pass_by_reference(string &v){
    string y =  "modified value pass_by_reference";
    v =y;
}
void pass_by_pointer(string *v){
    string y = "modified value pass_by_pointer";
    *v = y;
}
void pass_by_pointer_chage_pointing(string *v){
    string y = "modified value pass_by_pointer_chage_pointing";
    v = &y;
}

int main(){
    string x = "original value";
    string *ptr = &x;
    cout << x << endl;
    pass_by_value(x);
    cout << "x value after pass_by_value: " <<  x << endl;
    pass_by_reference(x);
    cout << "x value after pass_by_reference: " <<  x << endl;
    pass_by_pointer(ptr);
    cout << "x value after pass_by_pointer: " <<  x << endl;
    pass_by_pointer_chage_pointing(ptr);
    cout << "x value after pass_by_pointer_chage_pointing: " <<  x << endl;
    return 0;
}
```

If you can understand the above problem you have mastered pointers and references and this is one of the main use case of pointers and references.

Questions

Order of execution

You have to refer to the order of execution,[1]. C++ will do multple passes from left to right. Solve the example below, but when in double always use parenthesis as mentioned in the comment.

```
#include <stdio.h>
#include <iostream>
using namespace std;

int main(){
    double expresion = 600 / 10 + 30 + 100 % 6*6;
    /*
        (600 / 10) + (30) + (100 %6)*6;
    */

}
```

[1]https://en.cppreference.com/w/cpp/language/operator_precedence