



COLLEGE OF INFORMATICS AND VIRTUAL EDUCATION(CIVE)

Department of computer science and engineering

GROUP ASSIGNMENT:

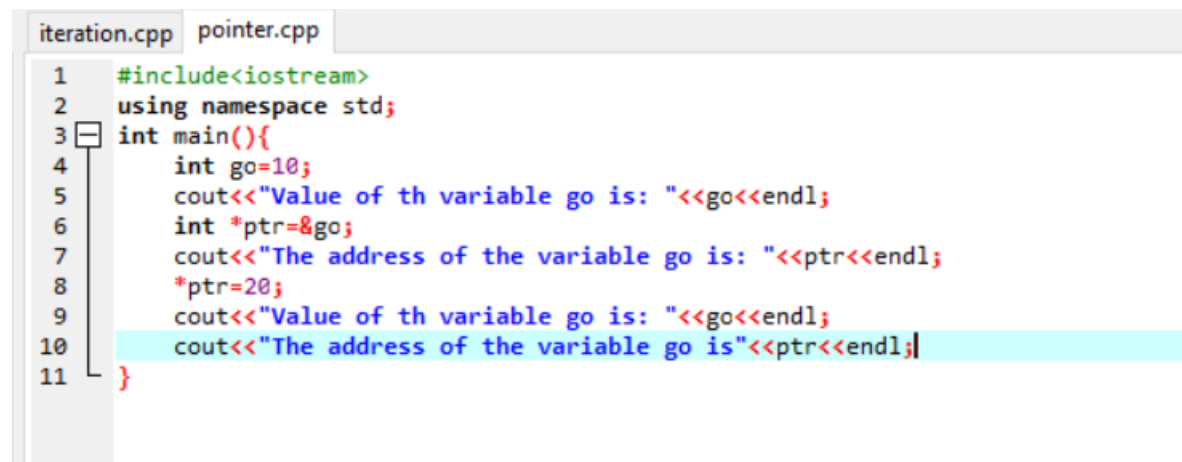
CLASS: BSc.SE

COURSE CODE: CP 213

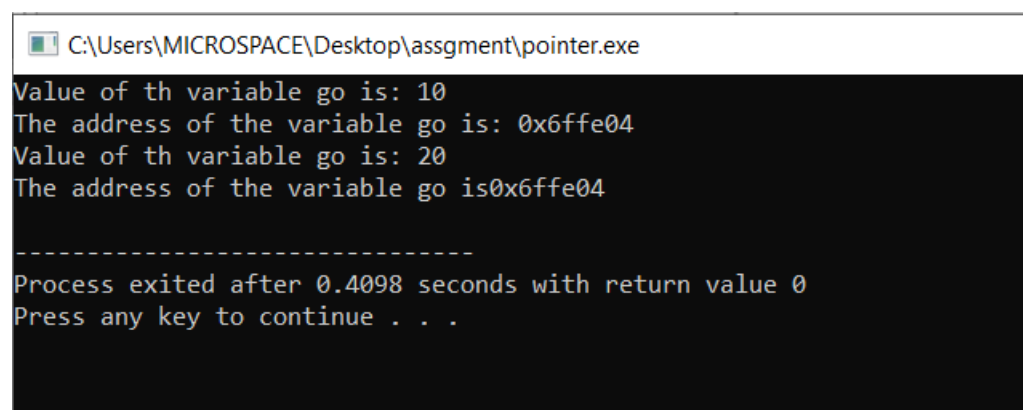
NAME	REGISTRATION NUMBER
SABRINA RAJAB MSELEM	T22-03-00737
NGASSA MASUMBUKO BENJAMIN	T22-03-09883
ELIA ELISHA MADUKWA	T22-03-11754
ERICK COSTER MBOYA	T22-03-06903
SALMIN NAGB SALMIN	T22-03-04324

Qn 1: Pointer is a variable that stores the memory address of another variable.

Meaning that it provides the location in the memory for specific value to be stored, it is declared using symbols like *, & with different meaning each with its meaning such as *x it assigns the value stored at the memory address. &x just initializes a pointer to the address of integer variable x.



```
1  #include<iostream>
2  using namespace std;
3  int main(){
4      int go=10;
5      cout<<"Value of th variable go is: "<<go<<endl;
6      int *ptr=&go;
7      cout<<"The address of the variable go is: "<<ptr<<endl;
8      *ptr=20;
9      cout<<"Value of th variable go is: "<<go<<endl;
10     cout<<"The address of the variable go is"<<ptr<<endl;
11 }
```



```
C:\Users\MICROSPACE\Desktop\assgment\pointer.exe
Value of th variable go is: 10
The address of the variable go is: 0x6ffe04
Value of th variable go is: 20
The address of the variable go is0x6ffe04

-----
Process exited after 0.4098 seconds with return value 0
Press any key to continue . . .
```

Qn 2: Static memory allocation occur when memory is allocated at compile time the size of the memory is determined before the program runs. The memory is typically allocated for variable declared at the beginning of the program or in a specific function

staticMemoryAllocation.cpp

```
1 #include<iostream>
2 using namespace std;
3 int main(){
4     //static memory allocation for variable
5     int y=10;
6     //static memory allocation for array variables
7     int staticAlloc[5]={4,6,8,9,5};
8     int sum=0;
9     for(int i=0;i<5;i++){
10         sum +=staticAlloc[i];
11     }
12     cout<<"sum is: "<<sum;
13
14
15     return 0;
16 }
```

C:\Users\MICROSPACE\Desktop\assgment\staticMemoryAllocation.exe

```
sum is: 32
-----
Process exited after 0.475 seconds with return value 0
Press any key to continue . . .
```

: Dynamic memory allocation it involves allocating memory at run time and the size of the allocated memory can be determined during program execution, and it's done using operator like 'new' and 'delete'. Where 'new' allow to add even during run time no matter the size you allocate before. 'delete' just help programmers to free the allocated memory .

dynamicAlloc.cpp

```
1 #include <iostream>
2 using namespace std;
3 int main() {
4     // Dynamic array with size determined at runtime
5     int size;
6     cout << "Enter the size of the dynamic array: ";
7     cin >> size;
8
9     // Dynamic memory allocation
10    int* dynamicArray = new int[size];
11
12    // Initializing elements
13    for (int i = 0; i < size; ++i) {
14        dynamicArray[i] = i + 1;
15    }
16
17    // Accessing elements
18    for (int i = 0; i < size; ++i) {
19        cout << dynamicArray[i] << " ";
20    }
21
22    //free the allocated memory
23    delete[] dynamicArray;
24
25    return 0;
26 }
```

```
Enter the size of the dynamic array: 6
1 2 3 4 5 6
-----
Process exited after 3.151 seconds with return value 0
Press any key to continue . . .
```

Qn 3: Static array just arrays which have fixed size which are determined during compile-time such they are declared with specific size, and the memory is allocated on the stack. In such a sense that are suitable when the size is constant and known beforehand

staticMemoryAllocation.cpp

```
1  #include<iostream>
2  using namespace std;
3  int main(){
4      //static memory allocation for variable
5      int y=10;
6      //static memory allocation for array variables
7      int staticAlloc[5]={4,6,8,9,5};
8      int sum=0;
9      for(int i=0;i<5;i++){
10         sum +=staticAlloc[i];
11     }
12     cout<<"sum is: "<<sum;
13
14
15     return 0;
16 }
```

C:\Users\MICROSPACE\Desktop\assgment\staticMemoryAllocation.exe

```
sum is: 32
-----
Process exited after 0.475 seconds with return value 0
Press any key to continue . . .
```

: Dynamic arrays are those having size that can be determined at runtime in such a way that they are created using pointers and memory allocation operators like 'new' and 'delete'. And are suitable when the size is not known at compile-time or when the size may change during program execution

```
iteration.cpp  pointer.cpp  dynamicAlloc.cpp

1  #include <iostream>
2  using namespace std;
3  int main() {
4      int size;
5      cout << "Enter the size of array ";
6      cin >> size;
7      int* dynamicArray = new int[size];
8      for (int i = 0; i < size; ++i) {
9          dynamicArray[i] = i + 1;
10     }
11     for (int i = 0; i < size; ++i) {
12         cout << dynamicArray[i] << " ";
13     }
14     delete[] dynamicArray;
15
16     return 0;
17 }
```

```
C:\Users\MICROSPACE\Desktop\assgment\dynamicAlloc.exe
Enter the size of array
2
1 2
-----
Process exited after 5.445 seconds with return value 0
Press any key to continue . . .
```

Qn 4: Iteration these involves repeated execution of set of statement or block of code. It is used for tasks that needs to be performed at a specific number of times or until a certain condition is met. We achieve it by a way o called looping which involves FOR, WHILE, DO WHILE. It is suitable for many tasks.

```
iteration.cpp
1  #include <iostream>
2  using namespace std;
3  int main() {
4      int X=0;
5      while(X<3){
6          cout<<X<<endl;
7          ++X;
8      }
9
10 }
```

```
C:\Users\MICROSPACE\Desktop\assgmet\iteration.exe
0
1
2
-----
Process exited after 0.04572 seconds with return value 0
Press any key to continue . . .
```

: Recursion as a programming concept where a function calls it self in order to solve a smaller instance of the same problem. It involves breaking down of the problem into smaller subproblems and solve them until reaching a base case, it requires careful design to ensure it converges to the base case and doesn't result in infinite recursion.

recursion.cpp

```
1  #include <iostream>
2  using namespace std;
3  int factorialNum(int num){
4      //base condition to stop function to run
5      if(num==0){
6          return 1;
7      }
8      return num * factorialNum(num-1);
9  }
10 int main() {
11     int x=4;
12     int a=factorialNum(x);
13     cout<<"Factorial of "<<x<<" is "<<a;
14 }
15
```

C:\Users\MICROSPACE\Desktop\assgment\recursion.exe

Factorial of 4 is 24

Process exited after 0.2147 seconds with return value 0
Press any key to continue . . .

Qn 5: A sentinel loop refers to the loop that continues to execute until a specific condition is met, typically involves the use of special value known as sentile value that signal the end of the loop. And are commonly used when the exact number of iterations is not known beforehand and depends on user input or external factors and just provide flexible way to handle varying amounts of input data while ensuring a clear termination condition.

```
sentinel.cpp
1  #include <iostream>
2  using namespace std;
3  int main() {
4      // Sentinel value to end the loop
5      int sentinel = -1;
6      int number;
7      int sum = 0;
8
9      // Sentinel Loop - continues until the sentinel value is entered
10     while (true) {
11         cout << "Enter a number (or -1 to end): ";
12         cin >> number;
13
14         // Check if the entered value is the sentinel value
15         if (number == sentinel) {
16             break; // Exit the loop if the sentinel value is entered
17         }
18
19         // Add the entered number to the sum
20         sum += number;
21     }
22
23     // Display the sum of the entered numbers
24     cout << "Sum of entered numbers: " << sum << endl;
25
26     return 0;
27 }
28
```

```
C:\Users\MICROSPACE\Desktop\assgment\sentinel.exe
Enter a number (or -1 to end):
2
Enter a number (or -1 to end):
3
Enter a number (or -1 to end):
6
Enter a number (or -1 to end):
-1
Sum of entered numbers: 11

-----
Process exited after 23.84 seconds with return value 0
Press any key to continue . . .
```

The program runs in such a way that when the condition met for value 2 it terminate but already provide the answer for the sum.