**LAB MANUAL**

**Functions**

**“Passing by Value”**

Programming Fundamentals

BS(DS/AI)

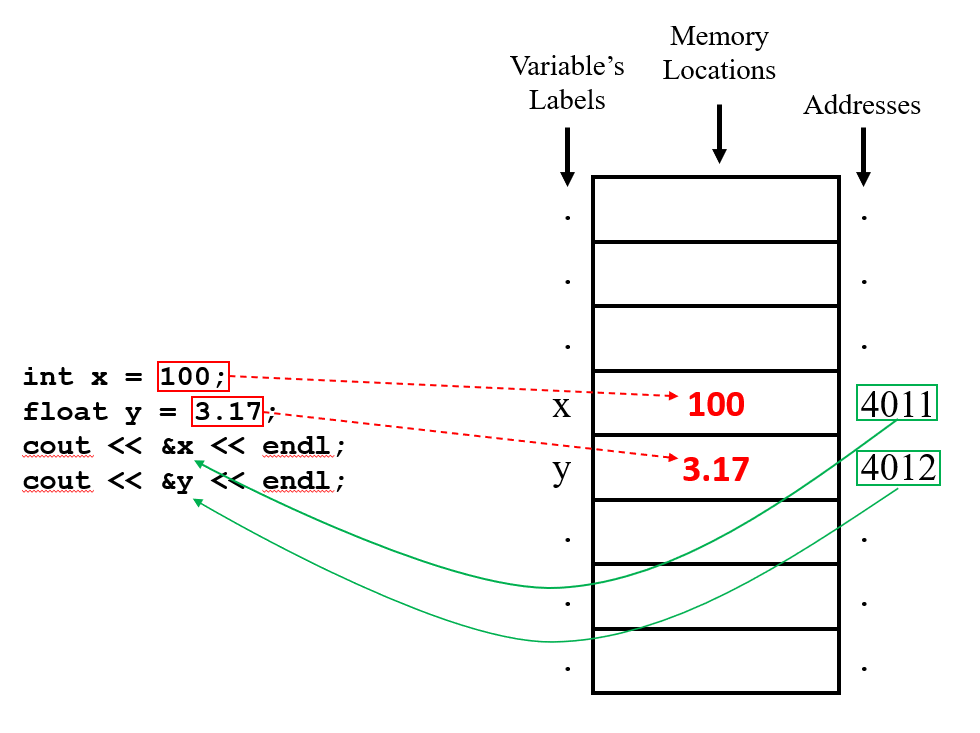


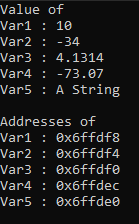
FAST National University of  
Computer and Emerging Sciences

**The Address-of Operator (Reference Declarator):**

* Ampersand sign ‘**&**’ is used as the address-of operator.
* Writing this operator with a variable will refer to the memory address of the variable.
* Address of variable ‘**y**’ is being stored in variable ‘**x**’ in the following example.

E.g. **x = &y**;



Example Program#01: Understanding the address operator in C++ (ampersand).

|  |  |
| --- | --- |
| 1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | #include<iostream>  #include<string>  **using** **namespace** std;  int main()  {  int Var1 = 10, Var2 = -34;  float Var3 = 4.1314, Var4 = -73.07;  string Var5 = "A String";  cout << "Value of" << endl  << "Var1 : " << Var1 << endl  << "Var2 : " << Var2 << endl  << "Var3 : " << Var3 << endl  << "Var4 : " << Var4 << endl  << "Var5 : " << Var5 << endl;    cout << "**\n**Addresses of" << endl  << "Var1 : " << &Var1 << endl  << "Var2 : " << &Var2 << endl  << "Var3 : " << &Var3 << endl  << "Var4 : " << &Var4 << endl  << "Var5 : " << &Var5 << endl;  } |

**Lab Task**

**Task1:**

Write a C++ function **isPerfectNumber** that checks if a number is a perfect number. A perfect number is a positive integer that is equal to the sum of its proper divisors, excluding the number itself. Function will return true if the number is perfect number otherwise return false.

*For example:*

6 is a perfect number because 1+2+3=6.

**Task 2:**

Write a C++ function **isArmStrong** that checks if a number is an Armstrong number. A number is an Armstrong number if the sum of its digits, each raised to the power of the number of digits, is equal to the number itself. Function will return true if the number is perfect number otherwise return false.

*For example:*

* 153 is an Armstrong number because 1^3 + 5^3 + 3^3 = 153
* 9474 is an Armstrong number because 9^4 + 4^4 + 7^4 + 4^4 = 9474.

**Task 3:**

Write a C++ function **findPrimeInRange** that finds and prints all prime numbers between two numbers. Function will accept two arguments **start** and **end.**

*For example:*

Start=10, end=50

11 13 17 19 23 29 31 37 41 43 47

**Task 4:**

Write a C++ function **fibonacciPosition** that returns the Fibonacci number at a given position in the sequence. The Fibonacci sequence is defined as:

F(0)=0, F(1)=1, F(n)=F(n−1)+F(n−2)

*For example:*

Input: num=7

Output: 13

**Submission instructions:**

* Submit single .cpp file containing code for all patterns.
* Rename your file as **i24XXXX\_LAB05.cpp** and then submit it on GCR.