# **CS 410 C++ to Assembly With Loops Activity Template**

**Step 1:** Explain the functionality of the C++ code.

## C++ Code Functionality

| **C++ Line of Code** | **Explanation of Functionality** |
| --- | --- |
| #include<iostream> | This is used to add in the iostream library. |
| Using namespace std; | Used to allow to code in standard way, no using std:: for each command. |
| Int main() | This is where the program begins its execution. |
| { |  |
| Int num, i; | Integer variables are made for num and I with nothing stored on them yet. |
| Int product = 1; | The integer variable for product is made and has the number 1 stored in it. |
| Cout<<”Enter a number:\n” << endl; | String is displayed for the user to read. |
| Cin >> num; | Input is requested to store an integer in the num variable. |
| for(i=num;i>0; i--) | A loop is created to run the sequence multiple times depending on the user’s variable input. I is assigned to be what ever the user input was. |
| product = product \* i; | The variable product is assigned by multiplying itself by I. |
| cout<<"The factorial for " << num << "is: \n"<< product; | Print out the factorial result according to the users input. |
| Return 1; | End program by returning 1. |

**Step 2:** Convert the C++ file into assembly code.

**Step 3:** Align each line of C++ code with the corresponding blocks of assembly code.

| C++ to Assembly Alignment**C++ Line of Code** | **Blocks of Assembly Code** |
| --- | --- |
| #include<iostream> |  |
| Using namespace std; |  |
| Int main() |  |
| { |  |
| Int num, i; |  |
| Int product = 1; | movl  $1, -12(%rbp) |
| Cout<<”Enter a number:\n” << endl; | leaq .LC0(%rip), %rsi    leaq  \_ZSt4cout(%rip), %rdi    call  \_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@PLT |
| Cin >> num; | leaq -20(%rbp), %rax    movq  %rax, %rsi    leaq  \_ZSt3cin(%rip), %rdi    call  \_ZNSirsERi@PLT |
| for(i=num;i>0; i--) | movl -20(%rbp), %eax    movl  %eax, -16(%rbp)    cmpl  $0, -16(%rbp)    jle .L2  subl $1, -16(%rbp)    jmp .L3 |
| product = product \* i; | movl  -12(%rbp), %eax    imull -16(%rbp), %eax    movl  %eax, -12(%rbp) |
| cout<<"The factorial for " << num << "is: \n"<< product; | leaq  .LC1(%rip), %rsi    leaq  \_ZSt4cout(%rip), %rdi    call  \_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@PLT    movq  %rax, %rdx    movl  -20(%rbp), %eax    movl  %eax, %esi    movq  %rdx, %rdi    call  \_ZNSolsEi@PLT    leaq  .LC2(%rip), %rsi    movq  %rax, %rdi    call  \_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@PLT    movq  %rax, %rdx    movl  -12(%rbp), %eax    movl  %eax, %esi    movq  %rdx, %rdi    call  \_ZNSolsEi@PLT |
| Return 1; | movl $1, %eax |

**Step 4:** Explain how the blocks of assembly code perform the same tasks as the C++ code.

## Assembly Functionality

| **Blocks of Assembly Code** | **Explanation of Functionality** |
| --- | --- |
| movl  $1, -12(%rbp) | This is assigning the value of 1 to a certain address of a variable, in this case -12(%rbp) is the location for the product integer. |
| leaq .LC0(%rip), %rsi    leaq  \_ZSt4cout(%rip), %rdi    call  \_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@PLT | LC0 is the location where the first string of the program is located and stored. The call function is then used to display the string. |
| leaq -20(%rbp), %rax    movq  %rax, %rsi    leaq  \_ZSt3cin(%rip), %rdi    call  \_ZNSirsERi@PLT | The call in this section of code is used to store what ever the user inputs into a certain location or variable. %rsi is the temporary location for the users input. |
| movl -20(%rbp), %eax    movl  %eax, -16(%rbp)    cmpl  $0, -16(%rbp)    jle .L2  subl $1, -16(%rbp)    jmp .L3 | This is the code for the for statement. JLE is code for jumping back to a specific location. These are done until the certain conditions are met in this case it would be when “I” reaches zero. |
| movl  -12(%rbp), %eax    imull -16(%rbp), %eax    movl  %eax, -12(%rbp) | This is code for multiplying variable -12(%rbp) (product) and the contents of cariable -16(%rbp) (I) and then after that they store the information back onto the products location. |
| leaq  .LC1(%rip), %rsi    leaq  \_ZSt4cout(%rip), %rdi    call  \_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@PLT    movq  %rax, %rdx    movl  -20(%rbp), %eax    movl  %eax, %esi    movq  %rdx, %rdi    call  \_ZNSolsEi@PLT    leaq  .LC2(%rip), %rsi    movq  %rax, %rdi    call  \_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@PLT    movq  %rax, %rdx    movl  -12(%rbp), %eax    movl  %eax, %esi    movq  %rdx, %rdi    call  \_ZNSolsEi@PLT | This is used to print out the done math and the users initial value that was passed into the program. LC1 is the location of the second string of the program that will just be displayed. It recalls location -20(%rbp) which is the initial num variable that the user assigned to display the number the started with. Then it uses LC2 to display the third string of the program that is stored in that location. Finally it called -12(rbp) which is the final product after the for loop exits. |
| movl $1, %eax | Exits program by returning 1. |