# **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> | Include the package iostream |
| using namespace std; | Defines scope of program |
| int main(){ | Declares main function |
| int num, i; | Declares 2 variables |
| int product =1; | Declares variable and sets it to 1 |
| cout<<"Enter a number:\n"<< endl; | Prints to terminal and new line |
| cin>>num; | Receives input and stores in variable |
| for(i=num;i>0; i--) | For loop and parameters for the loop |
| product = product \* i; | Multiplying a variable by itself and another variable |
| cout<<"The factorial for " << num << "is: \n"<< product; | Prints to terminal line a string and some variables |
| return 1;  } | Returns a 1 to let the computer know the program is over |

**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> | .file "assignment2\_1.cpp"  .text  .section .rodata  .type \_ZStL19piecewise\_construct, @object  .size \_ZStL19piecewise\_construct, 1 |
| using namespace std; |  |
| int main(){ | .globl main  .type main, @function  main: |
| int num, i; | .LFB1493:  .cfi\_startproc  pushq %rbp  .cfi\_def\_cfa\_offset 16  .cfi\_offset 6, -16  movq %rsp, %rbp  .cfi\_def\_cfa\_register 6  subq $32, %rsp  movq %fs:40, %rax  movq %rax, -8(%rbp)  xorl %eax, %eax  movl $1, -12(%rbp)  leaq .LC0(%rip), %rsi  leaq \_ZSt4cout(%rip), %rdi  call \_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@PLT  movq %rax, %rdx  movq \_ZSt4endlIcSt11char\_traitsIcEERSt13basic\_ostreamIT\_T0\_ES6\_@GOTPCREL(%rip), %rax  movq %rax, %rsi  movq %rdx, %rdi  call \_ZNSolsEPFRSoS\_E@PLT  leaq -20(%rbp), %rax  movq %rax, %rsi  leaq \_ZSt3cin(%rip), %rdi  call \_ZNSirsERi@PLT  movl -20(%rbp), %eax |
| int product =1; | .L3:  cmpl $0, -16(%rbp)  jle .L2  movl -12(%rbp), %eax  imull -16(%rbp), %eax  movl %eax, -12(%rbp)  subl $1, -16(%rbp)  jmp .L3  .L2:  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  movl $1, %eax  movq -8(%rbp), %rcx  xorq %fs:40, %rcx |
| cout<<"Enter a number:\n"<< endl; | .LC0:  .string "Enter a number:\n" |
| cin>>num; | je .L5  call \_\_stack\_chk\_fail@PLT  .L5:  leave  .cfi\_def\_cfa 7, 8  ret  .cfi\_endproc  .LFE1493:  .size main, .-main  .type \_Z41\_\_static\_initialization\_and\_destruction\_0ii, @function  \_Z41\_\_static\_initialization\_and\_destruction\_0ii: |
| for(i=num;i>0; i--) | .LFB1982:  .cfi\_startproc  pushq %rbp  .cfi\_def\_cfa\_offset 16  .cfi\_offset 6, -16  movq %rsp, %rbp  .cfi\_def\_cfa\_register 6  subq $16, %rsp  movl %edi, -4(%rbp)  movl %esi, -8(%rbp)  cmpl $1, -4(%rbp)  jne .L8  cmpl $65535, -8(%rbp)  jne .L8  leaq \_ZStL8\_\_ioinit(%rip), %rdi  call \_ZNSt8ios\_base4InitC1Ev@PLT  leaq \_\_dso\_handle(%rip), %rdx  leaq \_ZStL8\_\_ioinit(%rip), %rsi  movq \_ZNSt8ios\_base4InitD1Ev@GOTPCREL(%rip), %rax  movq %rax, %rdi  call \_\_cxa\_atexit@PLT  .L8:  nop  leave  .cfi\_def\_cfa 7, 8  ret  .cfi\_endproc  .LFE1982: |
| product = product \* i; | .LFB1983:  .cfi\_startproc  pushq %rbp  .cfi\_def\_cfa\_offset 16  .cfi\_offset 6, -16  movq %rsp, %rbp  .cfi\_def\_cfa\_register 6  movl $65535, %esi  movl $1, %edi  call \_Z41\_\_static\_initialization\_and\_destruction\_0ii  popq %rbp  .cfi\_def\_cfa 7, 8  ret  .cfi\_endproc  .LFE1983:s |
| cout<<"The factorial for " << num << "is: \n"<< product; | .LC1:  .string "The factorial for "  .LC2:  .string "is: \n" |
| return 1;  } | .LFE1493:  .size main, .-main  .type \_Z41\_\_static\_initialization\_and\_destruction\_0ii, @function |

**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** |
| --- | --- |
| .file "assignment2\_1.cpp"  .text  .section .rodata  .type \_ZStL19piecewise\_construct, @object  .size \_ZStL19piecewise\_construct, 1  \_ZStL19piecewise\_construct:  .zero 1  .local \_ZStL8\_\_ioinit  .comm \_ZStL8\_\_ioinit,1,1  .LC0:  .string "Enter a number:\n"  .LC1:  .string "The factorial for "  .LC2:  .string "is: \n"  .text  .globl main  .type main, @function | This block of code is declaring all of the objects and strings that are being used in the C program |
| main:  .LFB1493:  .cfi\_startproc  pushq %rbp  .cfi\_def\_cfa\_offset 16  .cfi\_offset 6, -16  movq %rsp, %rbp  .cfi\_def\_cfa\_register 6  subq $32, %rsp  movq %fs:40, %rax  movq %rax, -8(%rbp)  xorl %eax, %eax  movl $1, -12(%rbp)  leaq .LC0(%rip), %rsi  leaq \_ZSt4cout(%rip), %rdi  call \_ZStlsISt11char\_traitsIcEERSt13basic\_ostreamIcT\_ES5\_PKc@PLT  movq %rax, %rdx  movq \_ZSt4endlIcSt11char\_traitsIcEERSt13basic\_ostreamIT\_T0\_ES6\_@GOTPCREL(%rip), %rax  movq %rax, %rsi  movq %rdx, %rdi  call \_ZNSolsEPFRSoS\_E@PLT  leaq -20(%rbp), %rax  movq %rax, %rsi  leaq \_ZSt3cin(%rip), %rdi  call \_ZNSirsERi@PLT  movl -20(%rbp), %eax  movl %eax, -16(%rbp)  .L3:  cmpl $0, -16(%rbp)  jle .L2  movl -12(%rbp), %eax  imull -16(%rbp), %eax  movl %eax, -12(%rbp)  subl $1, -16(%rbp)  jmp .L3  .L2:  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  movl $1, %eax  movq -8(%rbp), %rcx  xorq %fs:40, %rcx  je .L5  call \_\_stack\_chk\_fail@PLT  .L5:  leave  .cfi\_def\_cfa 7, 8  ret  .cfi\_endproc  .LFE1493:  .size main, .-main  .type \_Z41\_\_static\_initialization\_and\_destruction\_0ii, @function  \_Z41\_\_static\_initialization\_and\_destruction\_0ii:  .LFB1982:  .cfi\_startproc  pushq %rbp  .cfi\_def\_cfa\_offset 16  .cfi\_offset 6, -16  movq %rsp, %rbp  .cfi\_def\_cfa\_register 6  subq $16, %rsp  movl %edi, -4(%rbp)  movl %esi, -8(%rbp)  cmpl $1, -4(%rbp)  jne .L8  cmpl $65535, -8(%rbp)  jne .L8  leaq \_ZStL8\_\_ioinit(%rip), %rdi  call \_ZNSt8ios\_base4InitC1Ev@PLT  leaq \_\_dso\_handle(%rip), %rdx  leaq \_ZStL8\_\_ioinit(%rip), %rsi  movq \_ZNSt8ios\_base4InitD1Ev@GOTPCREL(%rip), %rax  movq %rax, %rdi  call \_\_cxa\_atexit@PLT  .L8:  nop  leave  .cfi\_def\_cfa 7, 8  ret  .cfi\_endproc  .LFE1982:  .size \_Z41\_\_static\_initialization\_and\_destruction\_0ii, .-\_Z41\_\_static\_initialization\_and\_destruction\_0ii  .type \_GLOBAL\_\_sub\_I\_main, @function | This is the main function that takes the bytes and then moves them around the heap and stack to execute the commands from the c program.  The movq command moves all of the content from one location in memory to the next to keep up with what the c program is trying to do either adding or so on. The subq is from the sub command which is for subtraction. Althought there isn’t any subtraction in the c program it has to subtract from the memory to continue to move it. Then the call command is similar to a pointer and directs the flow of data. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |