# Elements of Computer Programming

ESC108A Elements of Computer Science and Engineering
B. Tech. 2017


**Course Leaders:**

**Roopa G.**

**Ami Rai E.**

**Chaitra S.**

# Objectives

- At the end of this lecture, student will be able to
  - identify categories of programming languages
  - identify the tools used for software development
  - explain the method of creating a computer program
  - use top-down approach to software development

# Contents

- Categories of Programming Languages
- Tools Used to Develop a Computer Program
- Execution of a C Program Using an IDE
- Top-Down Approach

# Programming

- Program
  - A well-defined set of instructions
- Programming
  - Process of writing instructions in a language for a computer to solve a specific task
- Programming languages
  - Medium of communication between the man and the machine

# Categories of Programming Languages

- Machine language
  - Easily understood by the machine
  - 0 and 1
  - Tedious for programmers
- Assembly Language
  - Mnemonics (“add”, “sub”, etc.)
- High Level Languages
  - High Level Language Instructions (closer to English)

# High Level Language - C

```c
/*
 * File:   Factorial.c
 * Author: vsarma
 * Created on 18 July, 2014, 12:11 PM
 */

#include <stdio.h>
#include <stdlib.h>
#define NUMBER 4/*The number for calculating factorial*/
/* Function fact:
 * Calculates factorial of a given number.
 * Input: A positive Integer number
 * Output: The factorial of the number or -1 in case of an error
 */
int fact(int n){
    int i;
    int ret = 1;
    if(n<0){
        return -1;
    }
    for(i=2;i<=n;i++)
    {
        ret = ret * i;
    }
    return ret;
}
```

# Assembly Language

```
fact:
.LFB0:
        .cfi_startproc
        pushl   %ebp
        .cfi_def_cfa_offset 8
        .cfi_offset 5, -8
        movl    %esp, %ebp
        .cfi_def_cfa_register 5
        subl    $16, %esp
        movl    $1, -8(%ebp)
        cmpl    $0, 8(%ebp)
        jns     .L2
        movl    $-1, %eax
        jmp     .L3
.L2:
        movl    $2, -4(%ebp)
        jmp     .L4
.L5:
        movl    -8(%ebp), %eax
        imull   -4(%ebp), %eax
        movl    %eax, -8(%ebp)
        addl    $1, -4(%ebp)
.L4:
        movl    -4(%ebp), %eax
        cmpl    8(%ebp), %eax
        jle     .L5
        movl    -8(%ebp), %eax
.L3:
        leave
        .cfi_restore 5
        .cfi_def_cfa 4, 4
        ret
        .cfi_endproc
.LFE0:
        .size   fact, .-fact
        .section        .rodata
.LC0:
        .string "Factorial of %d is %d\n"
        .text
        .globl  main
        .type   main, @function
```

# Machine Language

```
00000000  7f 45 4c 46 01 01 01 00  00 00 00 00 00 00 00 00  |.ELF............|
00000010  02 00 03 00 01 00 00 00  30 83 04 08 34 00 00 00  |........0...4...|
00000020  b8 0b 00 00 00 00 00 00  34 00 20 00 08 00 28 00  |........4. ...(.|
00000030  25 00 22 00 06 00 00 00  34 00 00 00 34 80 04 08  |%."....4...4...|
00000040  34 80 04 08 00 01 00 00  00 01 00 00 05 00 00 00  |4...............|
00000050  04 00 00 00 03 00 00 00  34 01 00 00 34 81 04 08  |........4...4...|
00000060  34 81 04 08 13 00 00 00  13 00 00 00 04 00 00 00  |4...............|
00000070  01 00 00 00 01 00 00 00  00 00 00 00 00 80 04 08  |................|
00000080  00 80 04 08 dc 05 00 00  dc 05 00 00 05 00 00 00  |................|
00000090  00 10 00 00 01 00 00 00  dc 05 00 00 dc 95 04 08  |................|
000000a0  dc 95 04 08 20 01 00 00  24 01 00 00 06 00 00 00  |.... ...$.......|
000000b0  00 10 00 00 02 00 00 00  e8 05 00 00 e8 95 04 08  |................|
000000c0  e8 95 04 08 f0 00 00 00  f0 00 00 00 06 00 00 00  |................|
000000d0  04 00 00 00 04 00 00 00  48 01 00 00 48 81 04 08  |........H...H...|
000000e0  48 81 04 08 44 00 00 00  44 00 00 00 04 00 00 00  |H...D...D.......|
000000f0  04 00 00 00 50 e5 74 64  38 05 00 00 38 85 04 08  |....P.td8...8...|
00000100  38 85 04 08 24 00 00 00  24 00 00 00 04 00 00 00  |8...$...$.......|
00000110  04 00 00 00 51 e5 74 64  00 00 00 00 00 00 00 00  |....Q.td........|
00000120  00 00 00 00 00 00 00 00  00 00 00 00 06 00 00 00  |................|
00000130  04 00 00 00 2f 6c 69 62  2f 6c 64 2d 6c 69 6e 75  |..../lib/ld-linu|
00000140  78 2e 73 6f 2e 32 00 00  04 00 00 00 10 00 00 00  |x.so.2..........|
00000150  01 00 00 00 47 4e 55 00  00 00 00 00 02 00 00 00  |....GNU.........|
00000160  06 00 00 00 1a 00 00 00  04 00 00 00 14 00 00 00  |................|
00000170  03 00 00 00 47 4e 55 00  6f 05 cc 96 62 0c 6e 10  |....GNU.o...b.n.|
00000180  20 7f eb c3 d8 c0 3a 7d  f4 7d be b3 03 00 00 00  | .....:}.}......|
00000190  05 00 00 00 02 00 00 00  03 00 00 00 04 00 00 00  |................|
000001a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000001b0  01 00 00 00 02 00 00 00  04 00 00 00 01 00 00 00  |................|
000001c0  05 00 00 00 00 20 00 20  00 00 00 00 04 00 00 00  |..... . ........|
000001d0  ad 4b e3 c0 00 00 00 00  00 00 00 00 00 00 00 00  |.K..............|
000001e0  00 00 00 00 29 00 00 00  00 00 00 00 00 00 00 00  |....)...........|
000001f0  12 00 00 00 01 00 00 00  00 00 00 00 00 00 00 00  |................|
00000200  20 00 00 00 30 00 00 00  00 00 00 00 00 00 00 00  | ...0...........|
00000210  12 00 00 00 1a 00 00 00  1c 85 04 08 04 00 00 00  |................|
00000220  11 00 10 00 00 5f 5f 67  6d 6f 6e 5f 73 74 61 72  |....._gmon_star|
00000230  74 5f 5f 00 6c 69 62 63  2e 73 6f 2e 36 00 5f 49  |t__.libc.so.6._I|
00000240  4f 5f 73 74 64 69 6e 5f  75 73 65 64 00 70 72 69  |O_stdin_used.pri|
```

# Tools Used to Develop a Computer Program

# Compiler/Interpreter

```c
/*
 * File:   Factorial.c
 * Author: vsarma
 * Created on 18 July, 2014, 12:11 PM
 */

#include <stdio.h>
#include <stdlib.h>
#define NUMBER 4/*The number for calculating factorial*/
/* Function fact:
 * Calculates factorial of a given number.
 * Input: A positive Integer number
 * Output: The factorial of the number or -1 in case of an error
 */
int fact(int n){
    int i;
    int ret = 1;
    if(n<0){
        return -1;
    }
    for(i=2;i<=n;i++)
    {
        ret = ret * i;
    }
    return ret;
}
```

Compiler / Interpreter

```
00000000  7f 45 4c 46 01 01 01 00  00 00 00 00 00 00 00 00  |.ELF............|
00000010  02 00 03 00 01 00 00 00  30 83 04 08 34 00 00 00  |........0...4...|
00000020  b8 0b 00 00 00 00 00 00  34 00 20 00 08 00 28 00  |........4. ...(.|
00000030  25 00 22 00 06 00 00 00  34 00 00 00 34 80 04 08  |%."....4...4...|
00000040  34 80 04 08 00 01 00 00  00 01 00 00 05 00 00 00  |4...............|
00000050  04 00 00 00 03 00 00 00  34 01 00 00 34 81 04 08  |........4...4...|
00000060  34 81 04 08 13 00 00 00  13 00 00 00 04 00 00 00  |4...............|
00000070  01 00 00 00 01 00 00 00  00 00 00 00 00 80 04 08  |................|
00000080  00 80 04 08 dc 05 00 00  dc 05 00 00 05 00 00 00  |................|
00000090  00 10 00 00 01 00 00 00  dc 05 00 00 dc 95 04 08  |................|
000000a0  dc 95 04 08 20 01 00 00  24 01 00 00 06 00 00 00  |.... ...$.......|
000000b0  00 10 00 00 02 00 00 00  e8 05 00 00 e8 95 04 08  |................|
000000c0  e8 95 04 08 f0 00 00 00  f0 00 00 00 06 00 00 00  |................|
000000d0  04 00 00 00 04 00 00 00  48 01 00 00 48 81 04 08  |........H...H...|
000000e0  48 81 04 08 44 00 00 00  44 00 00 00 04 00 00 00  |H...D...D.......|
000000f0  04 00 00 00 50 e5 74 64  38 05 00 00 38 85 04 08  |....P.td8...8...|
00000100  38 85 04 08 24 00 00 00  24 00 00 00 04 00 00 00  |8...$...$.......|
00000110  04 00 00 00 51 e5 74 64  00 00 00 00 00 00 00 00  |....Q.td........|
00000120  00 00 00 00 00 00 00 00  00 00 00 00 06 00 00 00  |................|
00000130  04 00 00 00 2f 6c 69 62  2f 6c 64 2d 6c 69 6e 75  |..../lib/ld-linu|
00000140  78 2e 73 6f 2e 32 00 00  04 00 00 00 10 00 00 00  |x.so.2..........|
00000150  01 00 00 00 47 4e 55 00  00 00 00 00 02 00 00 00  |....GNU.........|
00000160  06 00 00 00 1a 00 00 00  04 00 00 00 14 00 00 00  |................|
00000170  03 00 00 00 47 4e 55 00  6f 05 cc 96 62 0c 6e 10  |....GNU.o...b.n.|
00000180  20 7f eb c3 d8 c0 3a 7d  f4 7d be b3 03 00 00 00  | .....:}.}......|
00000190  05 00 00 00 02 00 00 00  03 00 00 00 04 00 00 00  |................|
000001a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000001b0  01 00 00 00 02 00 00 00  04 00 00 00 01 00 00 00  |................|
000001c0  05 00 00 00 00 20 00 20  00 00 00 00 04 00 00 00  |..... . ........|
000001d0  ad 4b e3 c0 00 00 00 00  00 00 00 00 00 00 00 00  |.K..............|
000001e0  00 00 00 00 29 00 00 00  00 00 00 00 00 00 00 00  |....}...........|
000001f0  12 00 00 00 01 00 00 00  00 00 00 00 00 00 00 00  |................|
00000200  20 00 00 00 30 00 00 00  00 00 00 00 00 00 00 00  | ...0...........|
00000210  12 00 00 00 1a 00 00 00  1c 85 04 08 04 00 00 00  |................|
00000220  11 00 10 00 00 5f 5f 67  6d 6f 6e 5f 73 74 61 72  |....._gmon_star|
00000230  74 5f 5f 00 6c 69 62 63  2e 73 6f 2e 36 00 5f 49  |t__.libc.so.6._I|
00000240  4f 5f 73 74 64 69 6e 5f  75 73 65 64 00 70 72 69  |O_stdin_used.pri|
```

# Compiler

- Translates High Level Language programs to Operational Codes at one go
- Compiled programs run faster as there is no translation during runtime
- Example: gcc (GNU C Compiler)

# Interpreter

# Interpreter

- Reads High Level Language programs line by line and executes their equivalent operational codes
- The process of interpretation makes the program execution slower
- Example:
  - Java Virtual Machine: Works on object code generated by Java compiler

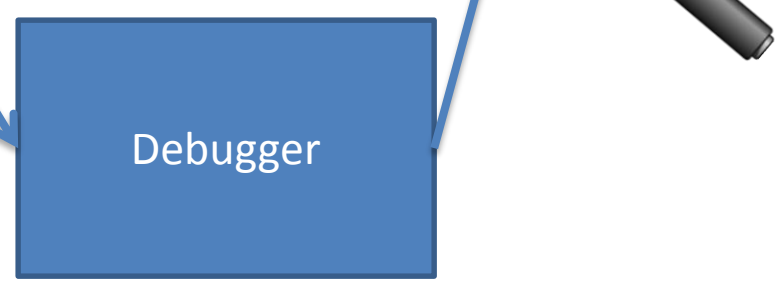
# Assembler

- Reads assembly language programs and translates into machine language
- Example:
  - TASM (Turbo ASseMbler)

# Debugger

```
00000000  7f 45 4c 46 01 01 01 00  00 00 00 00 00 00 00 00  |.ELF............|
00000010  02 00 03 00 01 00 00 00  30 83 04 08 34 00 00 00  |........0...4...|
00000020  b8 0b 00 00 00 00 00 00  34 00 20 00 08 00 28 00  |........4. ...(.|
00000030  25 00 22 00 06 00 00 00  34 00 00 00 34 80 04 08  |%."....4...4...|
00000040  34 80 04 08 00 01 00 00  00 01 00 00 05 00 00 00  |4...............|
00000050  04 00 00 00 03 00 00 00  34 01 00 00 34 81 04 08  |........4...4...|
00000060  34 81 04 08 13 00 00 00  13 00 00 00 04 00 00 00  |4...............|
00000070  01 00 00 00 01 00 00 00  00 00 00 00 00 80 04 08  |................|
00000080  00 80 04 08 dc 05 00 00  dc 05 00 00 05 00 00 00  |................|
00000090  00 10 00 00 01 00 00 00  dc 05 00 00 dc 95 04 08  |................|
000000a0  dc 95 04 08 20 01 00 00  24 01 00 00 06 00 00 00  |.... ...$.......|
000000b0  00 10 00 00 02 00 00 00  e8 05 00 00 e8 95 04 08  |................|
000000c0  e8 95 04 08 f0 00 00 00  f0 00 00 00 06 00 00 00  |................|
000000d0  04 00 00 00 04 00 00 00  48 01 00 00 48 81 04 08  |........H...H...|
000000e0  48 81 04 08 44 00 00 00  44 00 00 00 04 00 00 00  |H...D...D.......|
000000f0  04 00 00 00 50 e5 74 64  38 05 00 00 38 85 04 08  |....P.td8...8...|
00000100  38 85 04 08 24 00 00 00  24 00 00 00 04 00 00 00  |8...$...$.......|
00000110  04 00 00 00 51 e5 74 64  00 00 00 00 00 00 00 00  |....Q.td........|
00000120  00 00 00 00 00 00 00 00  00 00 00 00 06 00 00 00  |................|
00000130  04 00 00 00 2f 6c 69 62  2f 6c 64 2d 6c 69 6e 75  |..../lib/ld-linu|
00000140  78 2e 73 6f 2e 32 00 00  04 00 00 00 10 00 00 00  |x.so.2..........|
00000150  01 00 00 00 47 4e 55 00  00 00 00 00 02 00 00 00  |....GNU.........|
00000160  06 00 00 00 1a 00 00 00  04 00 00 00 14 00 00 00  |................|
00000170  03 00 00 00 47 4e 55 00  6f 05 cc 96 62 0c 6e 10  |....GNU.o...b.n.|
00000180  20 7f eb c3 d8 c0 3a 7d  f4 7d be b3 03 00 00 00  | .....:}.}......|
00000190  05 00 00 00 02 00 00 00  03 00 00 00 04 00 00 00  |................|
000001a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000001b0  01 00 00 00 02 00 00 00  04 00 00 00 01 00 00 00  |................|
000001c0  05 00 00 00 00 20 00 20  00 00 00 00 04 00 00 00  |..... . ........|
000001d0  ad 4b e3 c0 00 00 00 00  00 00 00 00 00 00 00 00  |.K..............|
000001e0  00 00 00 00 29 00 00 00  00 00 00 00 00 00 00 00  |....)...........|
000001f0  12 00 00 00 01 00 00 00  00 00 00 00 00 00 00 00  |................|
00000200  20 00 00 00 30 00 00 00  00 00 00 00 00 00 00 00  | ...0...........|
00000210  12 00 00 00 1a 00 00 00  1c 85 04 08 04 00 00 00  |................|
00000220  11 00 10 00 00 5f 5f 67  6d 6f 6e 5f 73 74 61 72  |....._gmon_star|
00000230  74 5f 5f 00 6c 69 62 63  2e 73 6f 2e 36 00 5f 49  |t__.libc.so.6._I|
00000240  4f 5f 73 74 64 69 6e 5f  75 73 65 64 00 70 72 69  |O_stdin_used.pri|
```

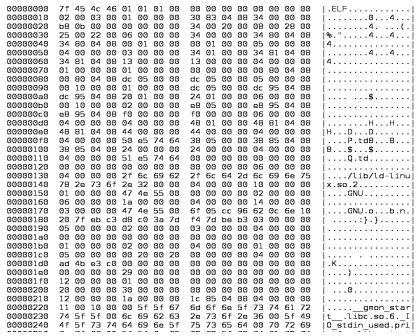
```
/*
 * File:   Factorial.c
 * Author: vsarma
 * Created on 18 July, 2014, 12:11 PM
 */

#include <stdio.h>
#include <stdlib.h>
#define NUMBER 4/*The number for calculating factorial*/
/* Function fact:
 * Calculates factorial of a given number.
 * Input: A positive Integer number
 * Output: The factorial of the number or -1 in case of an error
 */
int fact(int n){
    int i;
    int ret = 1;
    if(n<0){
        return -1;
    }
    for(i=2;i<=n;i++)
    {
        ret = ret * i;
    }
    return ret;
}
```

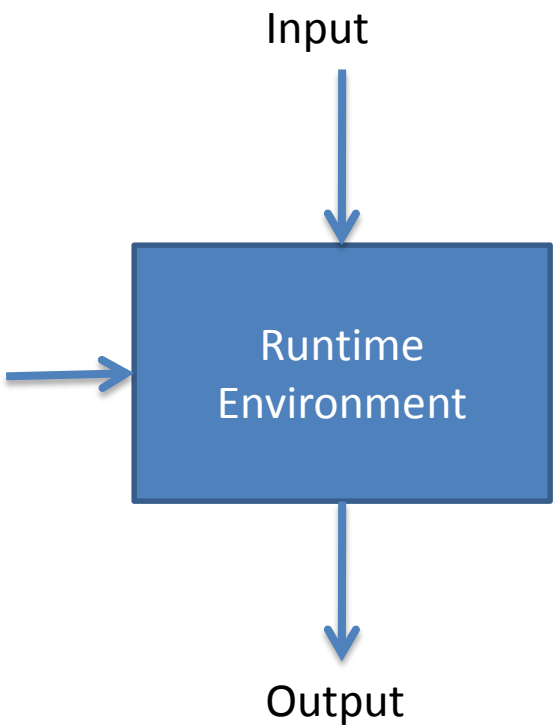


# Debugger

- Used for checking the run time condition of code
- Variable values during execution can be evaluated
- Useful in finding and removing ‘bugs’(errors in logic that cause unwanted results)
- Example: gdb (GNU Debugger)

# Runtime Environment






# Runtime Environment

- The environment where the compiled or interpreted code executes
- Consists of
  - a set of support libraries
  - operating system loader and scheduler

    or
  - a program that starts the execution of the developed program

# Runtime Environment

```
vsarma@vsarma-Desk-CE: ~/NetBeansPr
File  Edit  View  Search  Terminal  Help
vsarma@vsarma-Desk-CE:~/NetBeansProjects/Factorial$ locate libc.so.6
/lib/i386-linux-gnu/libc.so.6
/lib/i386-linux-gnu/i686/cmov/libc.so.6
vsarma@vsarma-Desk-CE:~/NetBeansProjects/Factorial$ gcc -o Factorial Factorial.c
vsarma@vsarma-Desk-CE:~/NetBeansProjects/Factorial$ ./Factorial
Factorial of 4 is 24
vsarma@vsarma-Desk-CE:~/NetBeansProjects/Factorial$
```

# Integrated Development Environment

- One tool that integrates compiler/interpreter, debugger and runtime environment in a user friendly interface
- Example: Netbeans

# Integrated Development Environment

Factorial - NetBeans IDE 8.0

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Projects × Files Services Classes

- Factorial
  - Header Files
  - Resource Files
  - Source Files
  - Test Files
  - Important Files

Navigator ×

- NUMBER
- fact(int n)
- main(int argc, char** argv)

Factorial.c ×

Source History

```c
 * Calculates factorial of a given number.
 * Input: A positive Integer number
 * Output: The factorial of the number or -1 in case of an error
 */
int fact(int n){
    int i;
    int ret = 1;
    if(n<0){
        return -1;
    }
    for(i=2;i<=n;i++)
    {
        ret = ret * i;
    }
    return ret;
}
/* Function main:
 * This is where the program starts.
 * This function calls fact function and prints the factorial of the NUMBER.
 */

int main(int argc, char** argv) {
    printf("Factorial of %d is %d\n",NUMBER,fact(NUMBER));
    return (EXIT_SUCCESS);
}
```

fact > for(i=2;i<=n;i++) >

Terminal - vsarma@vsarma-Desk-CE: ~ | Output | Variables × | Call Stack | Breakpoints

| Name | Value |
|---|---|
| <Enter new watch> | |
| i | 2 |
| ret | 1 |
| n | 4 |

User program stopped | Factorial (Debug) | 23:1 | INS

# Execution of a C Program Using IDE

- Clean and Build

Terminal - **vsarma@vsarma-Desk-CE: ~** | Output - CombinationProgram (Clean, Build)

```
"/usr/bin/make" -f nbproject/Makefile-Debug.mk QMAKE= SUBPROJECTS= .build-conf
make[1]: Entering directory `/home/vsarma/NetBeansProjects/CombinationProgram'
"/usr/bin/make"  -f nbproject/Makefile-Debug.mk dist/Debug/GNU-Linux-x86/combinationprogram
make[2]: Entering directory `/home/vsarma/NetBeansProjects/CombinationProgram'
mkdir -p build/Debug/GNU-Linux-x86
rm -f "build/Debug/GNU-Linux-x86/combination.o.d"
gcc    -c -g -MMD -MP -MF "build/Debug/GNU-Linux-x86/combination.o.d" -o build/Debug/GNU-Li
mkdir -p dist/Debug/GNU-Linux-x86
gcc     -o dist/Debug/GNU-Linux-x86/combinationprogram build/Debug/GNU-Linux-x86/combinatio
make[2]: Leaving directory `/home/vsarma/NetBeansProjects/CombinationProgram'
make[1]: Leaving directory `/home/vsarma/NetBeansProjects/CombinationProgram'

BUILD SUCCESSFUL (total time: 1s)
```

# Execution of a C Program Using IDE

- Run

Terminal - vsarma@vsarma-Desk-CE: ~ | Output ×

CombinationProgram (Clean, Build) × | CombinationProgram (Build, Run) × | CombinationProgram (Run) ×

```
"/usr/bin/make" -f nbproject/Makefile-Debug.mk QMAKE= SUBPROJECTS= .build-conf
make[1]: Entering directory `/home/vsarma/NetBeansProjects/CombinationProgram'
"/usr/bin/make"  -f nbproject/Makefile-Debug.mk dist/Debug/GNU-Linux-x86/combinationprogra
make[2]: Entering directory `/home/vsarma/NetBeansProjects/CombinationProgram'
make[2]: `dist/Debug/GNU-Linux-x86/combinationprogram' is up to date.
make[2]: Leaving directory `/home/vsarma/NetBeansProjects/CombinationProgram'
make[1]: Leaving directory `/home/vsarma/NetBeansProjects/CombinationProgram'

BUILD SUCCESSFUL (total time: 286ms)
```

# Execution of a C Program Using IDE

- Interacting with the program

Terminal - **vsarma@vsarma-Desk-CE: ~** | Output ×

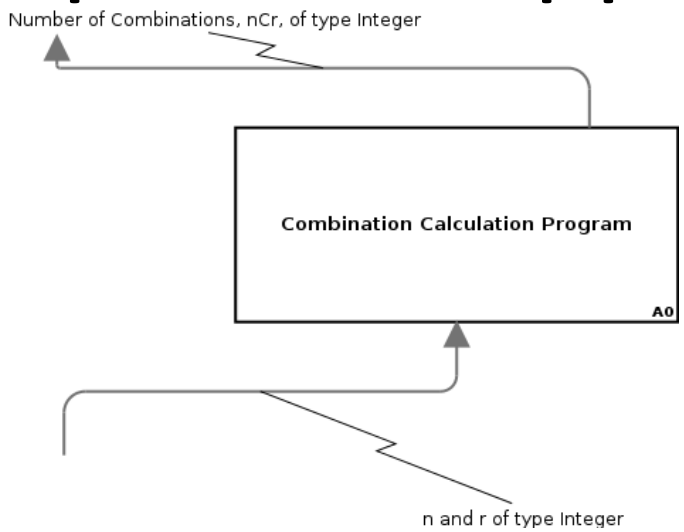
CombinationProgram (Clean, Build) × | CombinationProgram (Build, Run) × | CombinationProgram (Run) ×

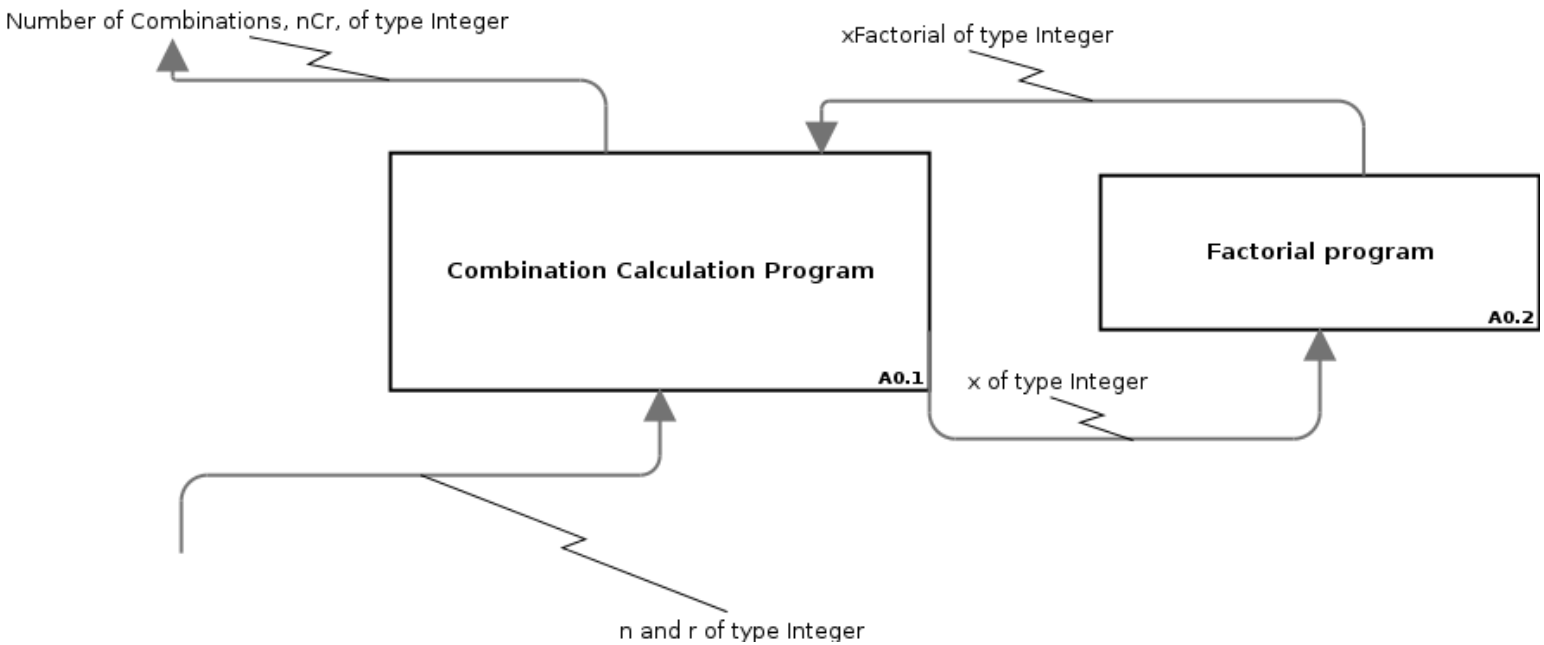
```
Please enter the values of n and r: 3,2
nCr is 3

RUN FINISHED; exit value 0; real time: 17s; user: 0ms; system: 0ms
```

# Top-Down Approach

- Identify what is given
  - Inputs and boundaries
- Identify what is expected
  - Outputs and error conditions
- Write steps to solve problem
  - Refine each step until further refinement is not possible
  - Identify sub problems and solve them independently

# Top-Down Approach



# Summary

- Computer understands only numbers
- A Compiler translates a high level language program in to an executable, while an interpreter translates it to machine code line by line
- De-bugger helps in troubleshooting the program by running the executable and allowing the user to control the execution and watch the values in the variables
- An IDE integrates all development tools and provides a simple interface
- Programs are developed in a top-down manner, first by understand the inputs and outputs, then by refining and then by expressing the logic as an algorithm