Computational Chemistry & Classical Molecular Dynamics

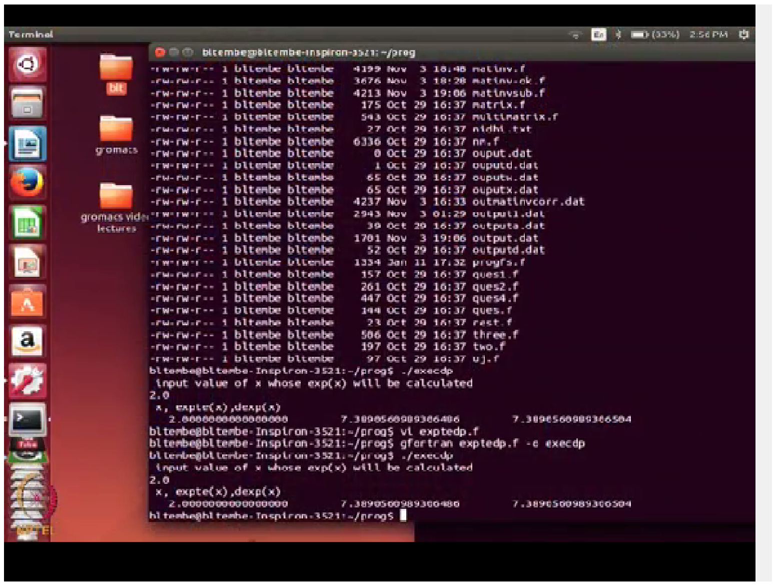
Prof. B. L. Tembe
Department of Chemistry
Indian Institute of Technology - Bombay

Lecture – 12
Practical Session on Programming 3: Functions and Subroutines

Hello and welcome again to this session in computational chemistry. So today we will be having our third practical session. In the last practical session, we executed an exponential program using a Taylor expansion. We calculated the exponential function using Taylor expansion and we took 250 terms in one calculation.

In another calculations, we took 350 terms and we also introduced the concept of double precision. What is double precision? I compute a number up to 16 decimal place accuracy, that is after the decimal point, I have 16 digits and my calculation is accurate up to those 16 digits. So we saw that our calculations did not make any difference whether we took 250 terms or 350 terms. In fact, that result you can pretty much see on the screen.

(Refer Slide Time: 01:12)



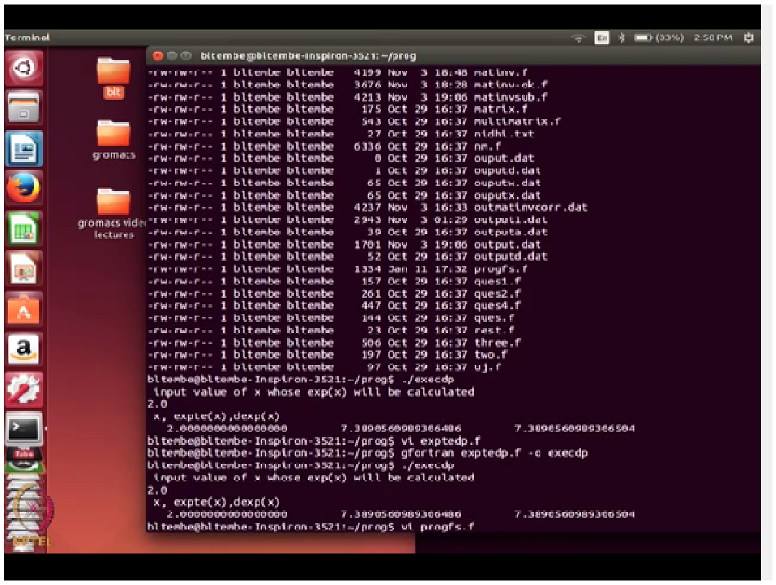
```
bltembe@bltembe-inspiron-3521: ~/prog
-rw-rw-r-- 1 bltembe bltembe   4199 Nov  3 18:48 matinv.f
-rw-rw-r-- 1 bltembe bltembe   3676 Nov  3 18:20 matinv-ok.f
-rw-rw-r-- 1 bltembe bltembe   4213 Nov  3 19:06 matinvsub.f
-rw-rw-r-- 1 bltembe bltembe    175 Oct 29 16:37 matrix.f
-rw-rw-r-- 1 bltembe bltembe    543 Oct 29 16:37 multimatrix.f
-rw-rw-r-- 1 bltembe bltembe     27 Oct 29 16:37 nidhi.txt
-rw-rw-r-- 1 bltembe bltembe   6336 Oct 29 16:37 nr.f
-rw-rw-r-- 1 bltembe bltembe      0 Oct 29 16:37 ouput.dat
-rw-rw-r-- 1 bltembe bltembe      1 Oct 29 16:37 ouputd.dat
-rw-rw-r-- 1 bltembe bltembe     65 Oct 29 16:37 ouputw.dat
-rw-rw-r-- 1 bltembe bltembe     65 Oct 29 16:37 ouputx.dat
-rw-rw-r-- 1 bltembe bltembe   4237 Nov  3 16:33 outmatinvcorr.dat
-rw-rw-r-- 1 bltembe bltembe   2943 Nov  3 01:29 outputi.dat
-rw-rw-r-- 1 bltembe bltembe     39 Oct 29 16:37 outputa.dat
-rw-rw-r-- 1 bltembe bltembe   1701 Nov  3 19:06 output.dat
-rw-rw-r-- 1 bltembe bltembe     52 Oct 29 16:37 outputd.dat
-rw-rw-r-- 1 bltembe bltembe   1334 Jan 11 17:32 progfs.f
-rw-rw-r-- 1 bltembe bltembe    157 Oct 29 16:37 ques1.f
-rw-rw-r-- 1 bltembe bltembe    261 Oct 29 16:37 ques2.f
-rw-rw-r-- 1 bltembe bltembe    447 Oct 29 16:37 ques4.f
-rw-rw-r-- 1 bltembe bltembe    144 Oct 29 16:37 ques.f
-rw-rw-r-- 1 bltembe bltembe     23 Oct 29 16:37 rest.f
-rw-rw-r-- 1 bltembe bltembe    506 Oct 29 16:37 three.f
-rw-rw-r-- 1 bltembe bltembe    197 Oct 29 16:37 two.f
-rw-rw-r-- 1 bltembe bltembe     97 Oct 29 16:37 uj.f
bltembe@bltembe-Inspiron-3521:~/prog$ ./execdp
 input value of x whose exp(x) will be calculated
2.0
 x, expte(x),dexp(x)
   2.0000000000000000        7.3890560989306486        7.3890560989306504
bltembe@bltembe-Inspiron-3521:~/prog$ vi exptedp.f
bltembe@bltembe-Inspiron-3521:~/prog$ gfortran exptedp.f -o execdp
bltembe@bltembe-Inspiron-3521:~/prog$ ./execdp
 input value of x whose exp(x) will be calculated
2.0
 x, expte(x),dexp(x)
   2.0000000000000000        7.3890560989306486        7.3890560989306504
bltembe@bltembe-Inspiron-3521:~/prog$
```

You see at the bottom of the screen. You can see that whatever we executed last time, my x value was 2.000. I have 2.00 written up to 16 digits. The calculated value as 7.389 up to 16 digits. The Fortran program's compiler also gave me 16 digits. So our digits, last 3 digits are 486. The Fortran compiler's result's are 504. We have an 18 digit difference between our result and Fortran result.

Now on the basis of our calculation, we can say that since 250 or 350 terms gave the same result, we can say that our results have converged and may be the Fortran compiler has used a different algorithm to calculate its exponential. Because remember any number can be calculated by different algorithms. Ours was a Taylor expansion algorithm. The computer compiler can use other algorithms to calculate this exponential function. So what we will do today now, we will execute the same programs using a function and a subroutine.

Remember, we have already discussed functions and subroutines. So what I will do in today's program is that have a main program which reads of lot of data, which prints data and it will call different functions and subroutines to execute the results that are required in the main program. So I have called it progfs, that is program which has function and subroutines. Let me edit that program so you will know what that program is going to do. So I will edit that program now.

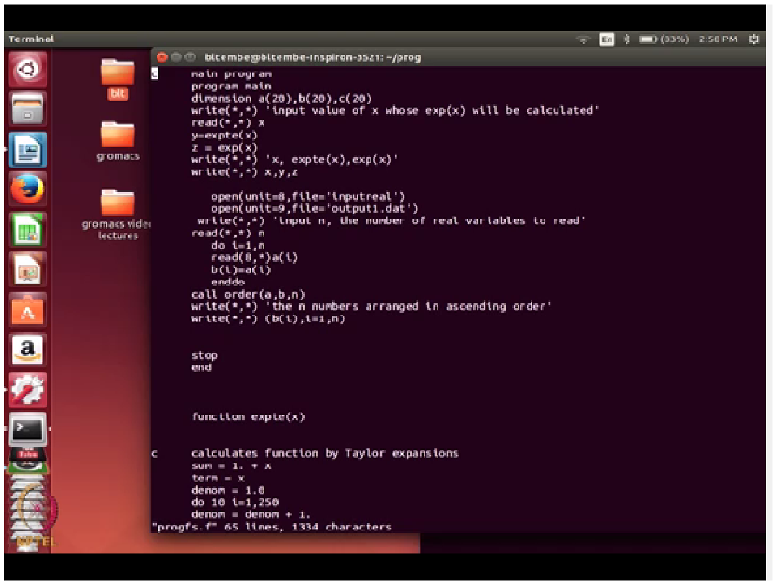
(Refer Slide Time: 02:48)



```
-rw-rw-r-- 1 bltembe bltembe   4199 Nov  3 18:48 matinv.f
-rw-rw-r-- 1 bltembe bltembe   3676 Nov  3 18:28 matinv-ok.f
-rw-rw-r-- 1 bltembe bltembe   4213 Nov  3 19:06 matinvsub.f
-rw-rw-r-- 1 bltembe bltembe    175 Oct 29 16:37 matrix.f
-rw-rw-r-- 1 bltembe bltembe    543 Oct 29 16:37 multimatrix.f
-rw-rw-r-- 1 bltembe bltembe     27 Oct 29 16:37 nidhi.txt
-rw-rw-r-- 1 bltembe bltembe   6336 Oct 29 16:37 nr.f
-rw-rw-r-- 1 bltembe bltembe      0 Oct 29 16:37 ouput.dat
-rw-rw-r-- 1 bltembe bltembe      1 Oct 29 16:37 ouputd.dat
-rw-rw-r-- 1 bltembe bltembe     65 Oct 29 16:37 ouputw.dat
-rw-rw-r-- 1 bltembe bltembe     65 Oct 29 16:37 ouputx.dat
-rw-rw-r-- 1 bltembe bltembe   4237 Nov  3 16:33 outmatinvcorr.dat
-rw-rw-r-- 1 bltembe bltembe   2943 Nov  3 01:29 output1.dat
-rw-rw-r-- 1 bltembe bltembe     39 Oct 29 16:37 outputa.dat
-rw-rw-r-- 1 bltembe bltembe   1701 Nov  3 19:06 output.dat
-rw-rw-r-- 1 bltembe bltembe     52 Oct 29 16:37 outputd.dat
-rw-rw-r-- 1 bltembe bltembe   1334 Jan 11 17:32 progfs.f
-rw-rw-r-- 1 bltembe bltembe    157 Oct 29 16:37 ques1.f
-rw-rw-r-- 1 bltembe bltembe    261 Oct 29 16:37 ques2.f
-rw-rw-r-- 1 bltembe bltembe    447 Oct 29 16:37 ques4.f
-rw-rw-r-- 1 bltembe bltembe    144 Oct 29 16:37 ques.f
-rw-rw-r-- 1 bltembe bltembe     23 Oct 29 16:37 rest.f
-rw-rw-r-- 1 bltembe bltembe    506 Oct 29 16:37 three.f
-rw-rw-r-- 1 bltembe bltembe    197 Oct 29 16:37 two.f
-rw-rw-r-- 1 bltembe bltembe     97 Oct 29 16:37 uj.f
bltembe@bltembe-Inspiron-3521:~/prog$ ./execdp
 input value of x whose exp(x) will be calculated
2.0
 x, expte(x),dexp(x)
   2.0000000000000000        7.3890560989306486        7.3890560989306504
bltembe@bltembe-Inspiron-3521:~/prog$ vi exptedp.f
bltembe@bltembe-Inspiron-3521:~/prog$ gfortran exptedp.f -o execdp
bltembe@bltembe-Inspiron-3521:~/prog$ ./execdp
 input value of x whose exp(x) will be calculated
2.0
 x, expte(x),dexp(x)
   2.0000000000000000        7.3890560989306486        7.3890560989306504
bltembe@bltembe-Inspiron-3521:~/prog$ vi progfs.f
```

How will I edit that program? Vi progfs.f. What this is doing?

(Refer Slide Time: 02:59)



```
c     main program
      program main
      dimension a(20),b(20),c(20)
      write(*,*) 'input value of x whose exp(x) will be calculated'
      read(*,*) x
      y=expte(x)
      z = exp(x)
      write(*,*) 'x, expte(x),exp(x)'
      write(*,*) x,y,z

         open(unit=8,file='inputreal')
         open(unit=9,file='output1.dat')
       write(*,*) 'input n, the number of real variables to read'
      read(*,*) n
         do i=1,n
         read(8,*)a(i)
         b(i)=a(i)
         enddo
      call order(a,b,n)
      write(*,*) 'the n numbers arranged in ascending order'
      write(*,*) (b(i),i=1,n)


      stop
      end



      function expte(x)


c     calculates function by Taylor expansions
      sum = 1. + x
      term = x
      denom = 1.0
      do 10 i=1,250
      denom = denom + 1.
"progfs.f" 65 lines, 1334 characters
```

This is a program in which there is a main program and there are functions as well as subroutines. So what this main program is going to do? It will call a function extex, that is a exponential function calculated using Taylor expansion exactly that we did earlier. See in the last class, we have calculated a function. But that was the whole program itself was the function.

Now in this particular calculation today, I have a main program and I have functions and subroutines, okay. So I will, there is 1 function and there is 1 subroutine. So we will execute that and we will also try to insert 1 more subroutine in this particular program. So that I shall do after I explain what all I have done so far. So what is the present program today? It calculates exponential function using a Taylor expansion through a function call statement, not in the main program.

Then the second part that this program will do, it will arrange numbers in an ascending order using a subroutine. So this particular program will do 2 tasks now and after I complete these 2 tasks, I will modify the program to calculate the product of 2 matrices. So let us go through this program line by line, okay. Now look at the starting point here. Now I have called the first line program main, okay.

So the Fortran compiler allows you to give certain name to your program. I have called it program main. And the next line is dimension a20 b20 c20. So what I have done? I have defined

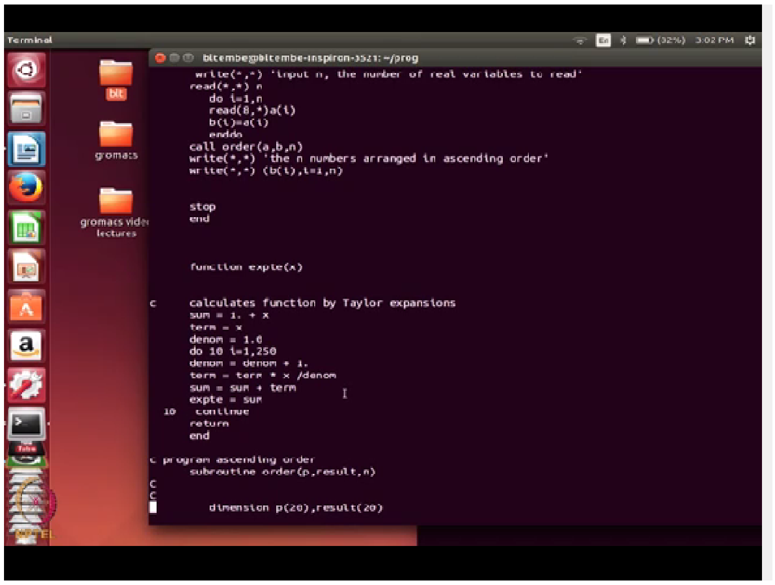
3 variables a, b and c. Both of these are dimension variables. What is the meaning of a dimension variable? That is a takes 20 values. What are those 20 values? a1 a2 a3 up to a20. b also takes 20 values.

c also takes 20 values. So Fortran compiler distinguishes between dimension variables and normal variables like x, y, z. When you do not say anything, the value x will take only 1 value but when you say a(20), so this variable a takes 20 values. So this is a dimension of 20. It is a single dimension. You could have double dimensions also. Like suppose I have a20, 20. So that will be a matrix.

So this is just an array. This is a vector, okay. A vector with 20 components. Just as scalars and vectors are different, dimension variables and undimension variables are different. So in a computer, your vector can have any number of dimensions. So I have declared my dimensions a, b, c. Next what I do I input the value of x whose exponential will be calculated, exactly as the previous program.

So once this line is executed, it will ask you to type x, read *,*x that means value of x will be read from the screen. So what is the next line? y=exptex, okay. So this y=exptex, it says that I have a function xte and that function will be calculated and that value will be given in the variable y. The moment I say y=exptex, the control of the program goes to this Fortran function. So remember once I type y=exptex, the control goes to function exptex.

(Refer Slide Time: 06:38)



```
      write(*,*) 'input n, the number of real variables to read'
      read(*,*) n
         do i=1,n
         read(8,*)a(i)
         b(i)=a(i)
         enddo
      call order(a,b,n)
      write(*,*) 'the n numbers arranged in ascending order'
      write(*,*) (b(i),i=1,n)


      stop
      end



      function expte(x)


c     calculates function by Taylor expansions
      sum = 1. + x
      term = x
      denom = 1.0
      do 10 i=1,250
      denom = denom + 1.
      term = term * x /denom
      sum = sum + term
      expte = sum
  10   continue
      return
      end

c program ascending order
      subroutine order(p,result,n)
c
c
         dimension p(20),result(20)
```

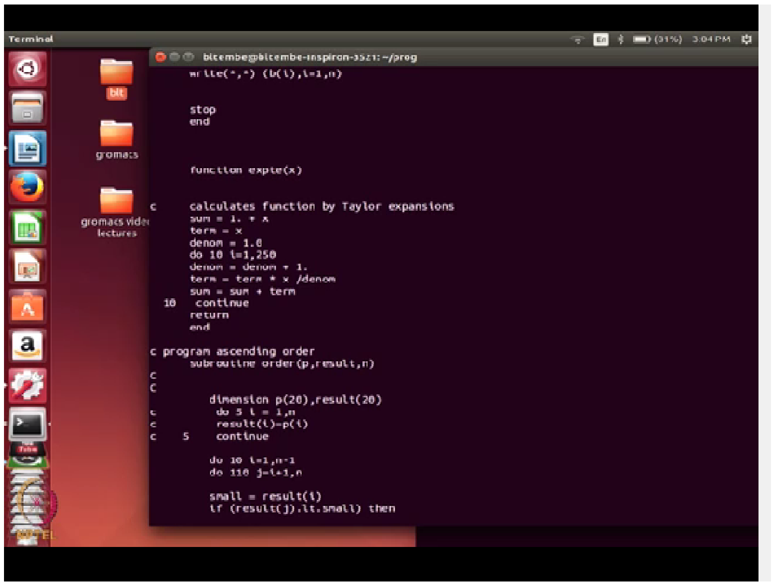
Now this exptex is exactly the program which we wrote last time. So what I did here? I copied the same function program into this program. I added this function exptex, whatever read statements I had here, remember I had read statement, let me go up and show you. So this statement write input value of x, this was in the main program, so I have brought it to main program.

Read x I have brought it to the main program. So the 2 input statements I have taken to the main program. Then the rest of it is the same, sum=1+x, term=x, denominator=1, do i going from 1 to 240, denominator=denominator+1, term=term*x/denom, all this part which was there in the main program, now I have brought it into this function value. And in addition to all this, there is a last statement, expte=sum.

So the name of the function is expte. So when it calculates all these things, whatever is the name of the function, the calculated value is put into that expte. So sum is calculated, it is put into expte. So here this expte is inside the do loop. Remember there is a do 10 i going from 1 to 250, all these statements are there.

This expte is inside this do loop. The correct way really is to put it outside the do loop. Because there is no point in defining expte again and again. So what I shall do now? I shall take this expte outside the do loop. How do I take it outside the do loop? I will go to that expte and I want to delete that line. So how do I delete the line?

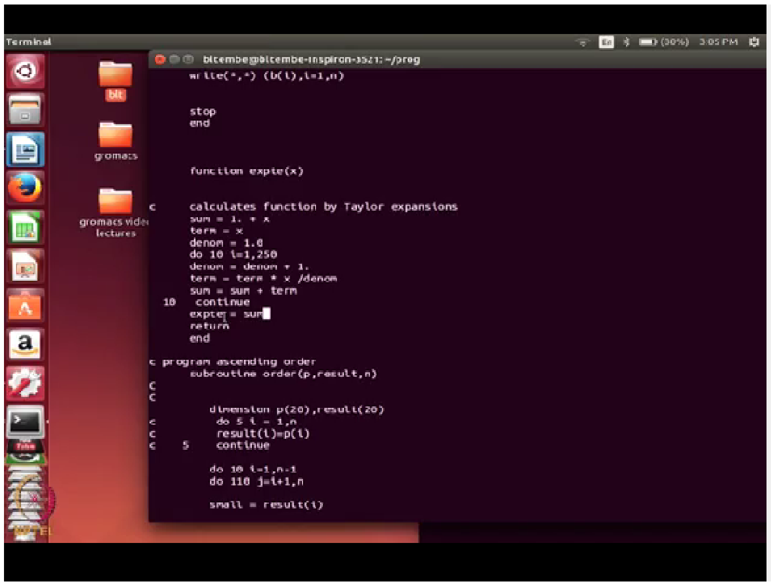
(Refer Slide Time: 08:26)



```
bitembe@bitembe-inspiron-3521: ~/prog
      write(*,*) (b(i),i=1,n)


      stop
      end



      function expte(x)


c     calculates function by Taylor expansions
      sum = 1. + x
      term = x
      denom = 1.0
      do 10 i=1,250
      denom = denom + 1.
      term = term * x /denom
      sum = sum + term
  10   continue
      return
      end

c program ascending order
      subroutine order(p,result,n)
c
C
         dimension p(20),result(20)
c         do 5 i = 1,n
c         result(i)=p(i)
c    5    continue

         do 10 i=1,n-1
         do 110 j=i+1,n

         small = result(i)
         if (result(j).lt.small) then
```

In vi editor, if you type dd twice, it will delete that line. So now I am going to type dd twice, that line is gone.

(Refer Slide Time: 08:32)

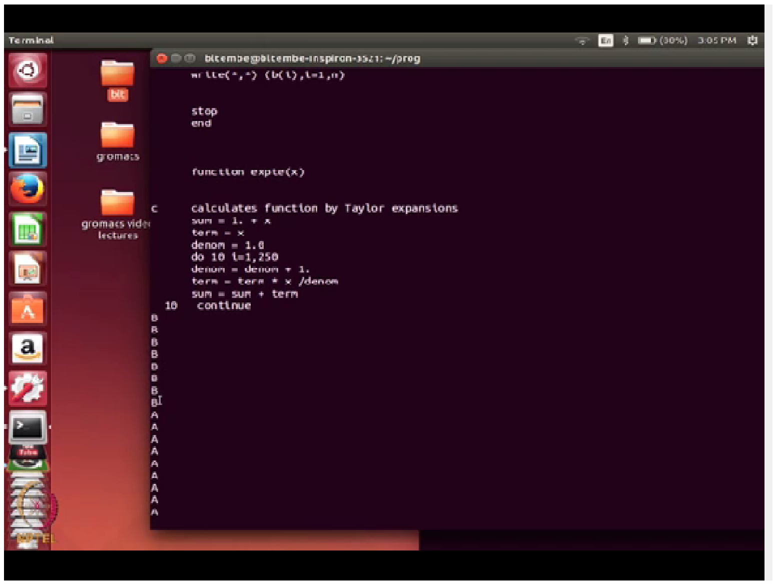


```
bitembe@bitembe-inspiron-3521: ~/prog
      write(*,*) (b(i),i=1,n)


      stop
      end



      function expte(x)


c     calculates function by Taylor expansions
      sum = 1. + x
      term = x
      denom = 1.0
      do 10 i=1,250
      denom = denom + 1.
      term = term * x /denom
      sum = sum + term
  10   continue
      expte = sum
      return
      end

c program ascending order
      subroutine order(p,result,n)
C
C
         dimension p(20),result(20)
c         do 5 i = 1,n
c         result(i)=p(i)
c    5    continue

         do 10 i=1,n-1
         do 110 j=i+1,n

         small = result(i)
```

So now I want to insert a line after 10 continue. So how do I do that? Now my cursor is at 1. So with my right arrow, I go all the way up to e, I am in the edit mode. Now I want to insert something after this line. I just press a. When I press a, I have gone to the right of e, then hit enter, then again 1 2 3 4 5 6, I leave the 6 places. Then I say expte = sum. Now what have I done. This particular line expte was inside the do loop.

See there is no need to define expte inside the loop because I am interested in the final value of the sum, okay. So therefore, one of the strategies in programming, whatever objects are to be calculated only once, they should never be inside the loop. So I have taken now expte outside the loop so that it is only calculated once after the entire sum is over. Then return and end. So when the function is called, it calculates all these things and returns the value of expte to the main program.

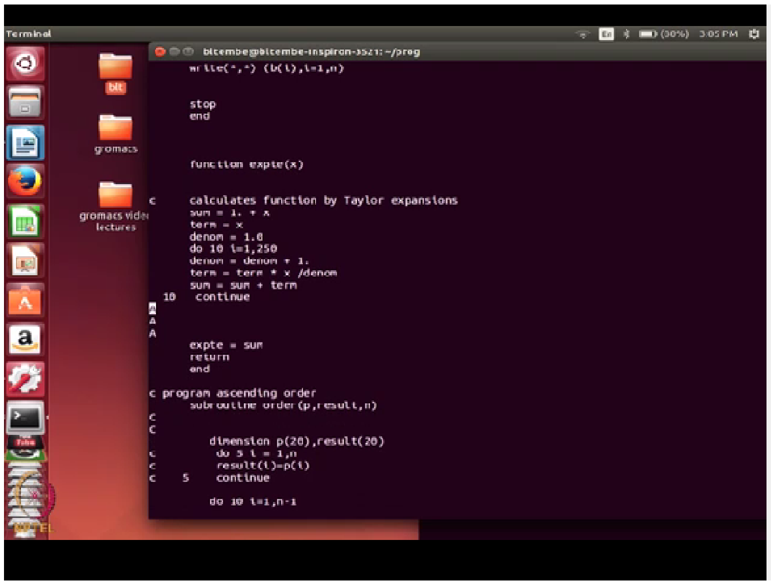
(Refer Slide Time: 09:44)



```
bitembe@bitembe-inspiron-3521: ~/prog
      write(*,*) (b(i),i=1,n)


      stop
      end



      function expte(x)


c     calculates function by Taylor expansions
      sum = 1. + x
      term = x
      denom = 1.0
      do 10 i=1,250
      denom = denom + 1.
      term = term * x /denom
      sum = sum + term
   10  continue
B
B
B
B
B
B
B
B
A
A
A
A
A
A
A
A
A
```

So remember now I did something wrong with my mouse, so it will insert all these lines. So I have to delete these lines. What do I do?

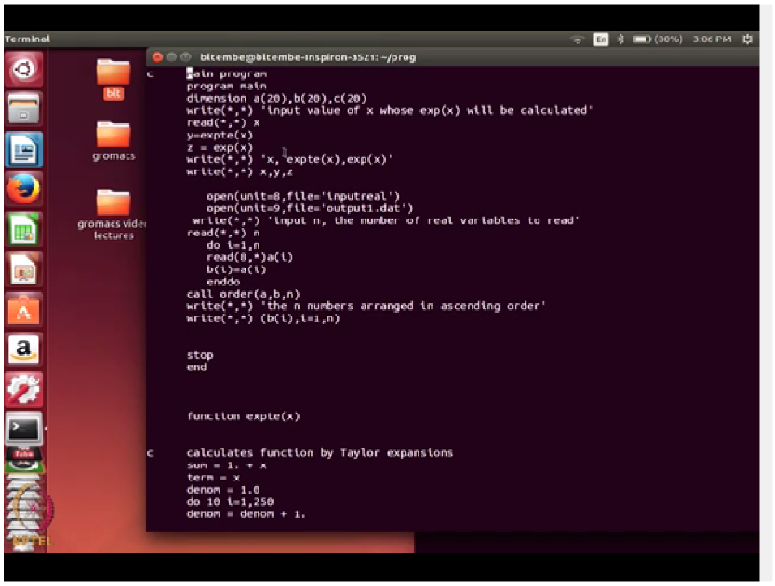
(Refer Slide Time: 09:52)



```
      write(*,*) (b(i),i=1,n)


      stop
      end



      function expte(x)


c     calculates function by Taylor expansions
      sum = 1. + x
      term = x
      denom = 1.0
      do 10 i=1,250
      denom = denom + 1.
      term = term * x /denom
      sum = sum + term
   10  continue
A
A
A


      expte = sum
      return
      end

c program ascending order
      subroutine order(p,result,n)
c
C
         dimension p(20),result(20)
c         do 5 i = 1,n
c         result(i)=p(i)
c    5    continue

         do 10 i=1,n-1
```

Escape, then I keep on typing dd twice. So I am, what I am doing? I am typing dd twice, I delete all the extra things I created, okay. So this vi is a very sensitive editor, okay. So if you just hit any character in the wrong place, it will give you problems. So whenever you insert, it should be either i and enter or a at the end of the line and enter, okay. Sum of that is one function part.

(Refer Slide Time: 10:23)



```
c     main program
      program main
      dimension a(20),b(20),c(20)
      write(*,*) 'input value of x whose exp(x) will be calculated'
      read(*,*) x
      y=expte(x)
      z = exp(x)
      write(*,*) 'x, expte(x),exp(x)'
      write(*,*) x,y,z

         open(unit=8,file='inputreal')
         open(unit=9,file='output1.dat')
       write(*,*) 'input n, the number of real variables to read'
      read(*,*) n
         do i=1,n
         read(8,*)a(i)
         b(i)=a(i)
         enddo
      call order(a,b,n)
      write(*,*) 'the n numbers arranged in ascending order'
      write(*,*) (b(i),i=1,n)


      stop
      end



      function expte(x)


c     calculates function by Taylor expansions
      sum = 1. + x
      term = x
      denom = 1.0
      do 10 i=1,250
      denom = denom + 1.
```

Now I want my program also to do one more thing. So I want to do my program to do 1 more thing that is arrange numbers in an ascending order. Remember we have already written a program to arrange numbers in an ascending order. Now I want to execute this ascending order program in a subroutine. So after it calculates the exponential of the function, y=exptex, z=expx, what does this z tells me?

This z is the value of the exponential calculated using the Fortran compiler. exptex is the value of the function calculated using our function which uses Taylor expansion. So both these are calculated, then these are written on the screen. So when I say write *,* x,y,z it writes the value of x which is my input from input value. y is the function calculated using Taylor expansion. z is the function calculated using the Fortran compiler.

So it will write all these things. Now I want my program to arrange numbers in an ascending order. So what do I have to do to arrange numbers in an ascending order? I have a subroutine to do that. So the first task that is needed is I need to read the input values. Input values of numbers that I want to arrange in an ascending order. So what I have done? I have created a data file. We will keep on needing lots of data in our execution in the future.

So I have created a file called inputreal.dat. You see this one. There is an input real, okay. This is the name of the file, inputreal. In that I have entered some 30 or 40 or 50 values of data. So whenever I need to read data, I will read from inputreal. So how do I connect my inputreal file to this program? It is through this open statement, open unit=8 file=inputreal. So this is an input file and then I will also create another file open unit=9 file=output1.dat.

So this is my file where I write all the results of my calculation, okay. So once I do that, what is my next line, write*,* input n, the number of real variables to read. So when I want to arrange numbers in an ascending order, I have to read given number of lines. So how many numbers I want to read? I want to read n variables. So I will write on the screen input the number of variables to be read and the next line is read*,*n.

What does the next line do? It will read n from the screen, okay. It will read n from the screen and when it reads n from the screen, what is my next thing? I have a do loop now. do i going from 1 to n. Read 8,*ai, so I am reading all the ai values from i going from 1 to n. Then the variable b is a second variable. So I want to call it bi=ai enddo. So now this do loop is slightly different from the do loops we have considered so far.

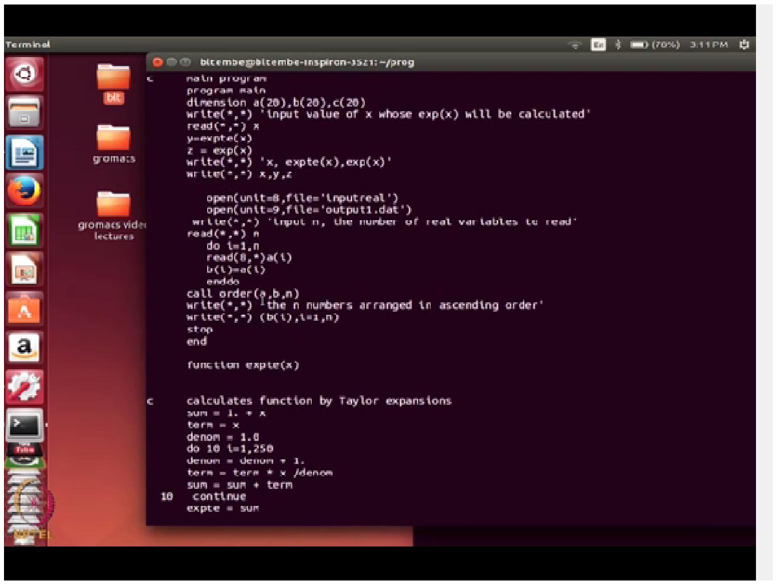
Earlier we use to give a line number for my do loop. This time what I have done? I have not used the line number. I have just used a do statement without line numbers. See lot of modern programming, they do not like too many line number. Because when you have to many line numbers, you go from 1 line to the next line. From that line to some other line. So you do not know exactly where the control will be.

So when you have lots of line numbers, and there is an error in the program, you will not be able to detect where the error occurs. So lesser the line numbers, the better because you will be able to identify the errors much better. So let us now go back to the program. So I have read all this data. I have made b=a, there is enddo. Then I say call order. So call order, whenever I have a call statement, it is subroutine.

So I say call order a,b,n. So I call that subroutine. Then what is the next thing? It goes to the subroutine, does the calculation and returns. So once it returns to the main program, write*,* the n numbers arranged in an ascending order. So if my program has worked, it would have created all my numbers in an ascending order. Then I will write it on the screen. I will write it on the screen.

Stop and end, okay. So you will see certain gaps here and certain gaps here. So the present compilers are such that a blank line does not affect the execution of the program. It is still good to avoid blank lines but I have put it so that you will be able to distinguish. So just for completeness, what I will do? Let me just delete a few blank lines so that there is no problem. So there are 2 blank lines.

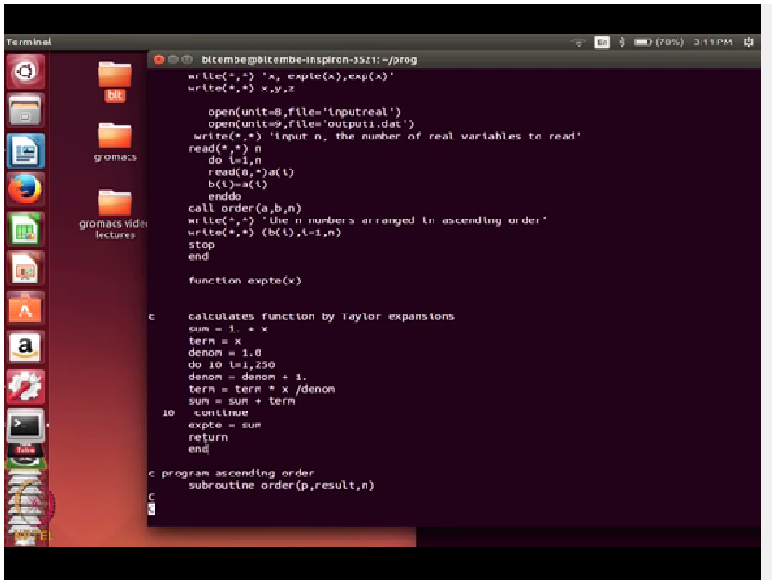
(Refer Slide Time: 15:20)



```
bitembe@bitembe-inspiron-3521: ~/prog
c     main program
      program main
      dimension a(20),b(20),c(20)
      write(*,*) 'input value of x whose exp(x) will be calculated'
      read(*,*) x
      y=expte(x)
      z = exp(x)
      write(*,*) 'x, expte(x),exp(x)'
      write(*,*) x,y,z

         open(unit=8,file='inputreal')
         open(unit=9,file='output1.dat')
       write(*,*) 'input n, the number of real variables to read'
      read(*,*) n
         do i=1,n
         read(8,*)a(i)
         b(i)=a(i)
         enddo
      call order(a,b,n)
      write(*,*) 'the n numbers arranged in ascending order'
      write(*,*) (b(i),i=1,n)
      stop
      end

      function expte(x)


c     calculates function by Taylor expansions
      sum = 1. + x
      term = x
      denom = 1.0
      do 10 i=1,250
      denom = denom + 1.
      term = term * x /denom
      sum = sum + term
 10    continue
      expte = sum
```

So I will, when I type d 2 times, it deletes it, okay. So it is always good to have some blank lines between some statements so that you can clearly distinguish one part from the other, okay. So this is the call statement to the order. Now let us see what the order subroutine is.

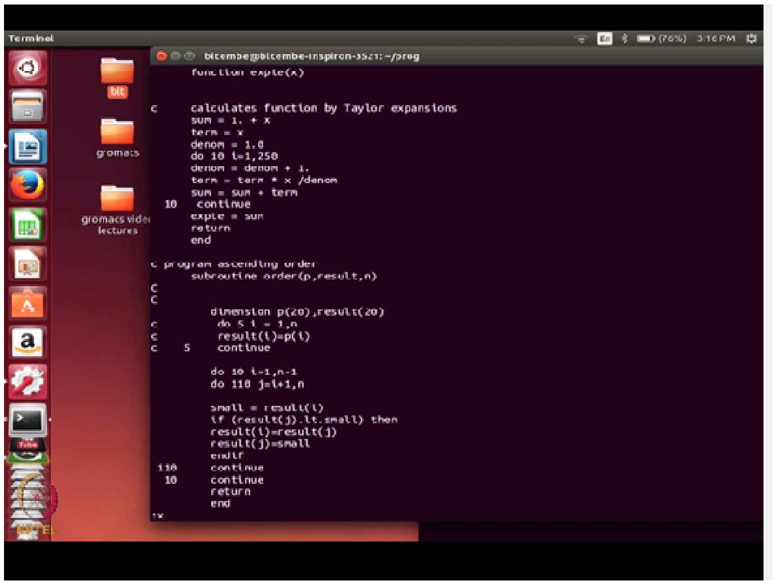
(Refer Slide Time: 15:38)



```
bitembe@bitembe-inspiron-3521: ~/prog
      write(*,*) 'x, expte(x),exp(x)'
      write(*,*) x,y,z

         open(unit=8,file='inputreal')
         open(unit=9,file='output1.dat')
       write(*,*) 'input n, the number of real variables to read'
      read(*,*) n
         do i=1,n
         read(8,*)a(i)
         b(i)=a(i)
         enddo
      call order(a,b,n)
      write(*,*) 'the n numbers arranged in ascending order'
      write(*,*) (b(i),i=1,n)
      stop
      end

      function expte(x)


c     calculates function by Taylor expansions
      sum = 1. + x
      term = x
      denom = 1.0
      do 10 i=1,250
      denom = denom + 1.
      term = term * x /denom
      sum = sum + term
 10    continue
      expte = sum
      return
      end

c program ascending order
      subroutine order(p,result,n)
c
c
```

So first I had written the function exptex. When I wrote this function part, there is a return and end. Remember in the main program, there is a stop and end. In the function, there is a return and end.

(Refer Slide Time: 15:53)



```
      function exple(x)


c     calculates function by Taylor expansions
      sum = 1. + x
      term = x
      denom = 1.0
      do 10 i=1,250
      denom = denom + 1.
      term = term * x /denom
      sum = sum + term
  10   continue
      exple = sum
      return
      end

c program ascending order
      subroutine order(p,result,n)
C
C
         dimension p(20),result(20)
c         do 5 i = 1,n
c         result(i)=p(i)
c    5    continue

         do 10 i=1,n-1
         do 110 j=i+1,n

         small = result(i)
         if (result(j).lt.small) then
         result(i)=result(j)
         result(j)=small
         endif
 110     continue
  10     continue
         return
         end
```

So after this, now I wrote the program for arranging numbers in an ascending order, okay. So this is the first. We wrote the program for the function. Next is the subroutine order. Now what is this now subroutine? I will call it subroutine order p,result,n. In the main program if you remember, I had a,b,n. So now I have p,result,n. So what does this do now? In my subroutine, the variable names are p, result and n.

In the main program, they have a, b and n. So it does not matter because whatever is the bracket between the main program and the subroutine, that is the common thing. In the main, I have called it a; in my subroutine, it is called p. In the main program, the second variable was b. In my case, in the subroutine, the second variable is result. n is the same. Instead of n, I could have called m.

It does not matter. So it also illustrates a very fundamental feature of this programming that the names in the subroutine can be all very different from the names in the main program. The line numbers in the subroutines can be any numbers. They will have nothing to do with the main program. The only thing that is linked between the main program and the subroutine is this whatever is passed through this brackets or whatever is passed through common statements.

Remember we have also discussed common statements. In the future, we will have a lot of occasion to use large number of common statements. And as I am executing the program, I would urge all of you to also put the computers on and execute it yourself. If you come across any difficulties or if you have any questions, you can always mail to us or to TA's here. We have 2 people to help in this course.

Our emails are given in our course introduction. So you can always contact us for any help you need in the executions because this is the theory cum practical course and you should execute every program that we are indicating here in our lectures. So now let us look at the subroutine order. Subroutine order, these are the subroutine. So I already told you that you match p, result and n to a, b and n in the main program.

So now we know that in the main program, a and b were array variables. They were not just normal variables, scalar variables. They were vector variables. What is the meaning of a vector variable? You will have more than 1 component. So both a and b in the main program have had 20 as their dimension. So in the subroutine as well, I will have dimension p20 and result 20. So instead of this 20, suppose I make it 21 or 19, there will be a huge error.

I would all urge you to purposely make it a different number then in the main program and see what error you will get. It may give the standard thing is some segmentation for all kinds of errors you will get if there is any mistake. So I have defined the dimension statement. So now just as in the main program, I wrote whatever was in a, I had written in b. So here also I do the same thing.

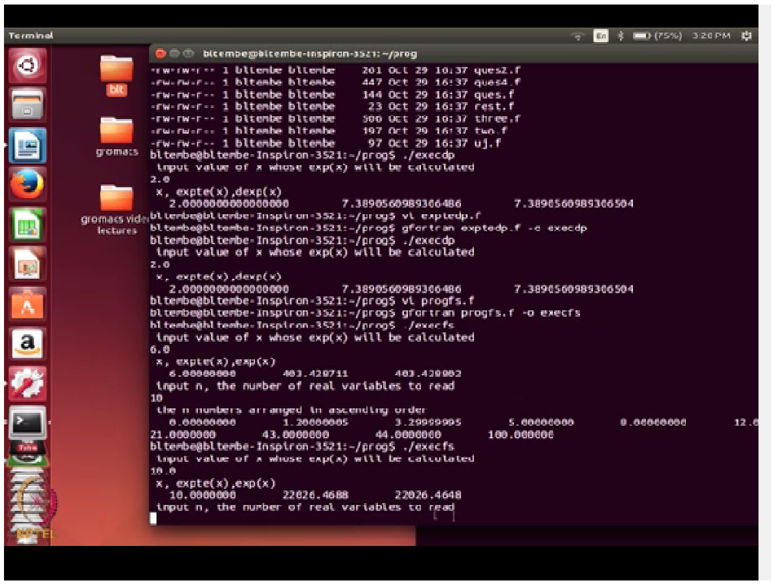
Whatever was in variable p, I have put it in result. So result array will be exactly as the p array because all the calculations I want to do on the result array and not on the initial p array which is the initial order. P has the initial data, result will have the final data. So in this do loop, my result is equated to pi. Now see the remaining lines. What are the remaining lines? So there are 2 do loops now, i going from, do 10 i going from 1 to n-1.

Second loop, j going from i+1 to n. So how do I exchange numbers? Remember what is the strategy? So whenever you want to exchange result i and result j, you first define a new temporary variable called small. Small is equated to result i and then if result j is less than small, then I exchange result i and result j. If result i is < small, then result i=result j and result j is small because small is the earlier value of result i.

So this small is necessary because if I do not use this small, then both result I and result j will have the same value of result j. So this if statement now, if some condition is satisfied, then you do these 2 lines and endif. So this is now a block statement. This if statement is a block statement. What is the meaning of that?

Whatever is the condition in the if statement, if that is satisfied, all the lines following that if statement up to the endif will be executed, okay. So in this case I execute this. Then 110 continue, 10 continue, return end. So this is the end of my subroutine. Remember again I have a return and end statement. So I shall now come out of this. How do I come out? Escape shift :x. x means it will save the program.

(Refer Slide Time: 21:09)



```
bltembe@bltembe-Inspiron-3521: ~/prog
-rw-rw-r-- 1 bltembe bltembe    201 Oct 29 16:37 ques2.f
-rw-rw-r-- 1 bltembe bltembe    447 Oct 29 16:37 ques4.f
-rw-rw-r-- 1 bltembe bltembe    144 Oct 29 16:37 ques.f
-rw-rw-r-- 1 bltembe bltembe     23 Oct 29 16:37 rest.f
-rw-rw-r-- 1 bltembe bltembe    500 Oct 29 16:37 three.f
-rw-rw-r-- 1 bltembe bltembe    197 Oct 29 16:37 two.f
-rw-rw-r-- 1 bltembe bltembe     97 Oct 29 16:37 uj.f
bltembe@bltembe-Inspiron-3521:~/prog$ ./execdp
 input value of x whose exp(x) will be calculated
2.0
 x, expte(x),dexp(x)
   2.0000000000000000        7.3890560989306486        7.3890560989306504
bltembe@bltembe-Inspiron-3521:~/prog$ vi exptedp.f
bltembe@bltembe-Inspiron-3521:~/prog$ gfortran exptedp.f -o execdp
bltembe@bltembe-Inspiron-3521:~/prog$ ./execdp
 input value of x whose exp(x) will be calculated
2.0
 x, expte(x),dexp(x)
   2.0000000000000000        7.3890560989306486        7.3890560989306504
bltembe@bltembe-Inspiron-3521:~/prog$ vi progfs.f
bltembe@bltembe-Inspiron-3521:~/prog$ gfortran progfs.f -o execfs
bltembe@bltembe-Inspiron-3521:~/prog$ ./execfs
 input value of x whose exp(x) will be calculated
6.0
 x, expte(x),exp(x)
   6.00000000        403.428711        403.428802
 input n, the number of real variables to read
10
 the n numbers arranged in ascending order
   0.00000000        1.20000005        3.29999995        5.00000000        8.00000000        12.0
21.0000000        43.0000000        44.0000000        100.000000
bltembe@bltembe-Inspiron-3521:~/prog$ ./execfs
 input value of x whose exp(x) will be calculated
10.0
 x, expte(x),exp(x)
   10.0000000        22026.4688        22026.4648
 input n, the number of real variables to read
```

Now I will compile this. What is my present program? Program is programfs.f. What is this program? It has subroutines as well as functions. So gfortran prog function subroutine.f. So I am compiling. So let us give it a new name. So what name I want to give? -o so the name I want to give it would be, let us say exec, I will call it execfs, because in this particular program, I am executing functions and subroutines.

So my new name is execfs. So the compiled version will be putting execfs, enter. So now it has compiled. So I will execute. To execute, I will say ./execfs. So it will execute this program. So I enter. So what is the first thing it asks me? It is asking me the value of x whose exponential it wants to calculate. So let me now give a large value, 6.0. So 6.0 is a value. So now just, okay, so I will enter.

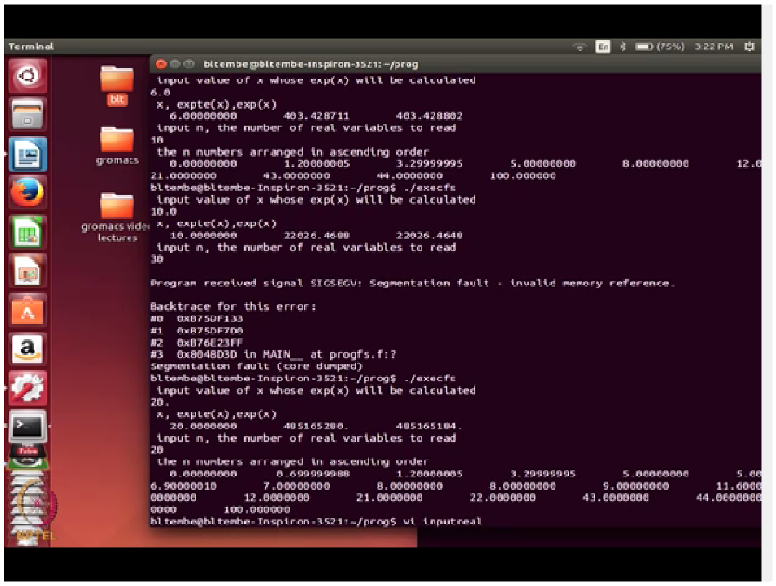
So it has given the values now. 6 is the value, 403.42871, 1 is the calculation in single precision now. Because we do not have a real*8 in this program. So these are the values calculated from our algorithm, Taylor expansion whereas from the Fortran compiler 403.428802. Remember now there is a lot of difference between our calculation and the compiler's calculation, okay. So now we have not investigated which is better but that we can always do as an exercise.

But now what I want to do after executing this function? Now I want to arrange numbers in an ascending order. Remember that my data is already in a file called inputreal. So it now asks me inputn, the number of real variables to read. So let me give 10 as the number now. So I want to read 10, so that means it will read 10 numbers from that file and execute the arrangement in an ascending order and it will print my results on the screen. I enter.

Now see, these are all the values. These are the 10 values, 0 1.2 3.29 5 8 12 21 43 44 100. So it has read the data from my file and arranged them in an ascending order. Now let us try to execute it again for one more set of data. So again execfs./execfs enter. So this time instead of 6, I want to give a value 10.0. So that means I want to calculate the exponential of 10, enter.

So what it has done now? 22026.4688 22026.4648. So now the difference now was in the third place after the decimal, 464, this is 468, so this has calculated the exponential of this larger number now. Now I want to arrange numbers in an ascending order. Last time I used 10 numbers. So let me use 30 numbers now.

(Refer Slide Time: 24:48)

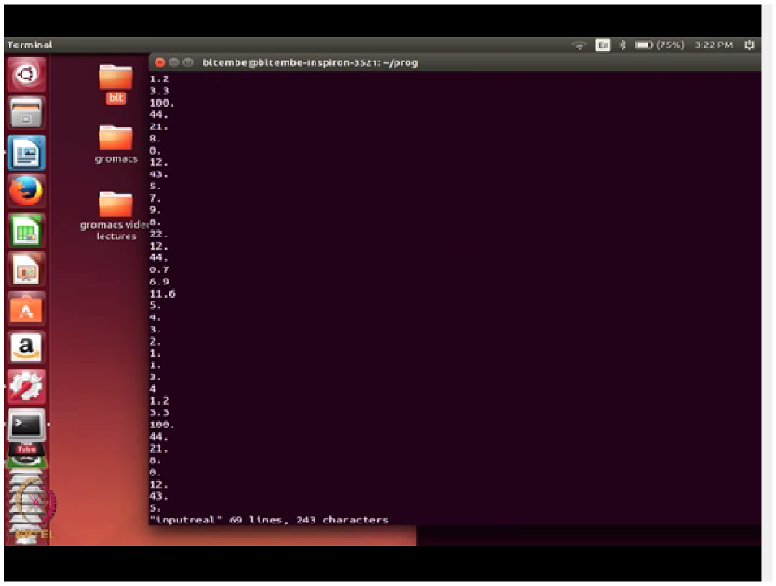


```
 input value of x whose exp(x) will be calculated
6.0
 x, expte(x),exp(x)
   6.00000000       403.428711       403.428802
 input n, the number of real variables to read
10
 the n numbers arranged in ascending order
   0.00000000       1.20000005       3.29999995       5.00000000       8.00000000       12.0
21.0000000       43.0000000       44.0000000       100.000000
bltembe@bltembe-Inspiron-3521:~/prog$ ./execfs
 input value of x whose exp(x) will be calculated
10.0
 x, expte(x),exp(x)
   10.0000000       22026.4600       22026.4648
 input n, the number of real variables to read
30

Program received signal SIGSEGV: Segmentation fault - invalid memory reference.

Backtrace for this error:
#0  0XB75DF133
#1  0xB75DF7D0
#2  0xB76E23FF
#3  0x8048D3D in MAIN__ at progfs.f:?
Segmentation fault (core dumped)
bltembe@bltembe-Inspiron-3521:~/prog$ ./execfs
 input value of x whose exp(x) will be calculated
20.
 x, expte(x),exp(x)
   20.0000000       485165280.       485165184.
 input n, the number of real variables to read
20
 the n numbers arranged in ascending order
   0.00000000       0.699999988       1.20000005       3.29999995       5.00000000       5.00
6.90000010       7.00000000       8.00000000       8.00000000       9.00000000       11.0000
0000000       12.0000000       21.0000000       22.0000000       43.0000000       44.0000000
0000       100.000000
bltembe@bltembe-Inspiron-3521:~/prog$ vi inputreal
```

Because I told you I have a data file with lots of data. So I will read 30 real numbers from that file and arrange them in an ascending order, see whether it works. I enter. So now it is saying, so there is some problem with this, okay. So the reason was, so it has a problem of reading 30 numbers. So let us try for some other data, okay. So I had used x=30. Now let me use x=20. It calculated exponential of 20, e to the 20, 485165280 485165184.

You already see that for large values of x, the difference is between our Taylor expansion and the Fortran compiler's values are different. Now I want to arrange numbers in an ascending order. Last time I gave 30 I will give 20, okay. So it has excepted these 20 numbers. See it has arranged 0 69 1.2 3 5, it has arranged them all in an ascending order. Now we saw that it gave some error when I gave 30 as the number of data to be read. So to understand that, let us go to that file. vi inputreal.

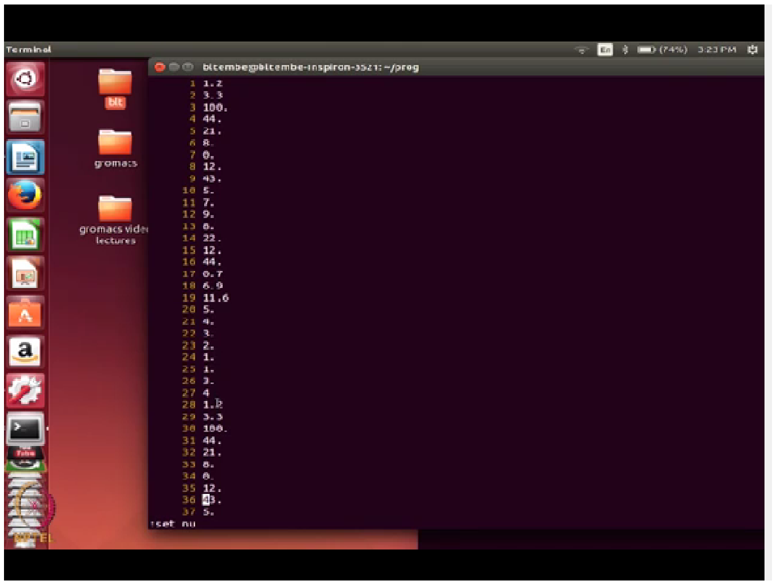
(Refer Slide Time: 26:30)



```
bitembe@bitembe-inspiron-3521: ~/prog
1.2
3.3
100.
44.
21.
8.
0.
12.
43.
5.
7.
9.
0.
22.
12.
44.
0.7
6.9
11.6
5.
4.
3.
2.
1.
1.
3.
4
1.2
3.3
100.
44.
21.
8.
0.
12.
43.
5.
"inputreal" 69 lines, 243 characters
```

So these are all my data. So let us see whether there is any problem in this. So first one was 1.2, 3.3, 100, 44. So there has to be some error in this data. Otherwise, it will not stop, okay. On the face of it, I do not see any error here, okay. So now my file has lots of data here. So I want to number these lines. So there is a Linux command to number these lines, shift : What I did?

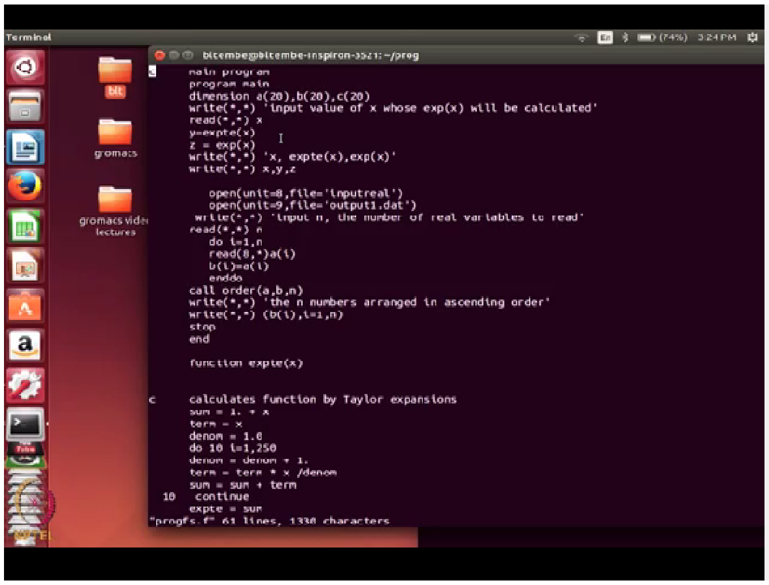
(Refer Slide Time: 27:10)



```
bitembe@bitembe-inspiron-3521: ~/prog
      1 1.2
      2 3.3
      3 100.
      4 44.
      5 21.
      6 8.
      7 0.
      8 12.
      9 43.
     10 5.
     11 7.
     12 9.
     13 0.
     14 22.
     15 12.
     16 44.
     17 0.7
     18 6.9
     19 11.6
     20 5.
     21 4.
     22 3.
     23 2.
     24 1.
     25 1.
     26 3.
     27 4
     28 1.2
     29 3.3
     30 100.
     31 44.
     32 21.
     33 8.
     34 0.
     35 12.
     36 43.
     37 5.
:set nu
```

I was editing this file inputreal, shift :, then I typed when the colon appears at the bottom, set nu. So it sets the numbers to these lines, enter. So you see what this has done? It has given line numbers to all my programs, okay. So 30 data I had given. So I do not see any major problem here in these numbers. So why it should give an error, I have no clue. So I do not see any major problem in my data. So let me try to execute again to see whether I can do it.

(Refer Slide Time: 27:59)

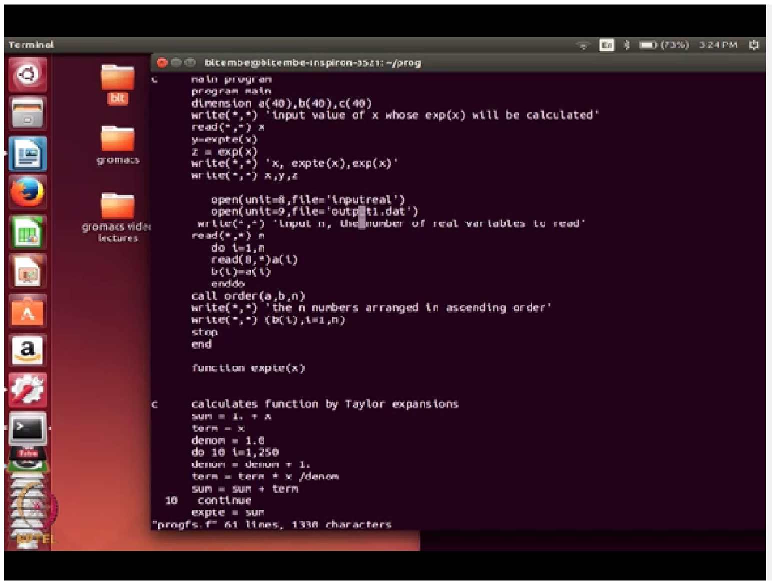


```
bitembe@bitembe-inspiron-3521: ~/preg
c     main program
      program main
      dimension a(20),b(20),c(20)
      write(*,*) 'input value of x whose exp(x) will be calculated'
      read(*,*) x
      y=expte(x)
      z = exp(x)
      write(*,*) 'x, expte(x),exp(x)'
      write(*,*) x,y,z

         open(unit=8,file='inputreal')
         open(unit=9,file='output1.dat')
       write(*,*) 'input n, the number of real variables to read'
      read(*,*) n
         do i=1,n
         read(8,*)a(i)
         b(i)=a(i)
         enddo
      call order(a,b,n)
      write(*,*) 'the n numbers arranged in ascending order'
      write(*,*) (b(i),i=1,n)
      stop
      end

      function expte(x)


c     calculates function by Taylor expansions
      sum = 1. + x
      term = x
      denom = 1.0
      do 10 i=1,250
      denom = denom + 1.
      term = term * x /denom
      sum = sum + term
  10   continue
      expte = sum
"progfs.f" 61 lines, 1330 characters
```

Okay, now I understood the problem, okay. So let us see, escape. Let me edit that program and then you will know what is the error is. vi progfs.f. So remember my array sizes were 20. My array sizes were 20 and I tried to read 30 numbers. So there is no way you can read 30 numbers when the array size is 20. So it tries to read more than what the size can accommodate.

It is like if you have 72 seats in a train and you try to occupy 150 people, it cannot accept it. Of course, you have a train, any number of people will go. That is a different situation. But in a program, if you have 20 spaces in an array, it cannot accommodate 21 or 22. And since I tried to do 30 times, it gave a problem, okay. So if I want to do this, one way to do would be, I change, okay, we will do this.

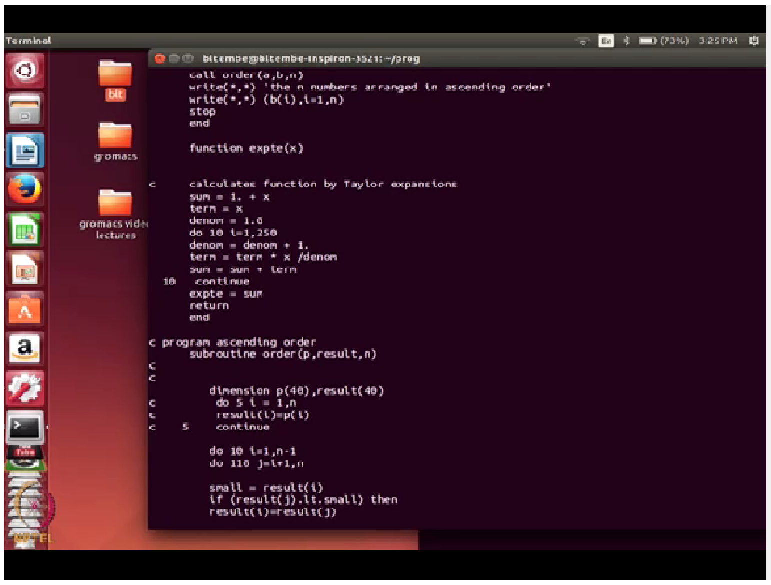
(Refer Slide Time: 28:58)



```
bitembe@bitembe-inspiron-3521: ~/prog
c     main program
      program main
      dimension a(40),b(40),c(40)
      write(*,*) 'input value of x whose exp(x) will be calculated'
      read(*,*) x
      y=expte(x)
      z = exp(x)
      write(*,*) 'x, expte(x),exp(x)'
      write(*,*) x,y,z

         open(unit=8,file='inputreal')
         open(unit=9,file='output1.dat')
       write(*,*) 'input n, the number of real variables to read'
      read(*,*) n
         do i=1,n
         read(8,*)a(i)
         b(i)=a(i)
         enddo
      call order(a,b,n)
      write(*,*) 'the n numbers arranged in ascending order'
      write(*,*) (b(i),i=1,n)
      stop
      end

      function expte(x)


c     calculates function by Taylor expansions
      sum = 1. + x
      term = x
      denom = 1.0
      do 10 i=1,250
      denom = denom + 1.
      term = term * x /denom
      sum = sum + term
   10  continue
      expte = sum
"progfs.f" 61 lines, 1330 characters
```

So what I will do? So I will make all my arrays 40, okay. I made them 40 in my main program.

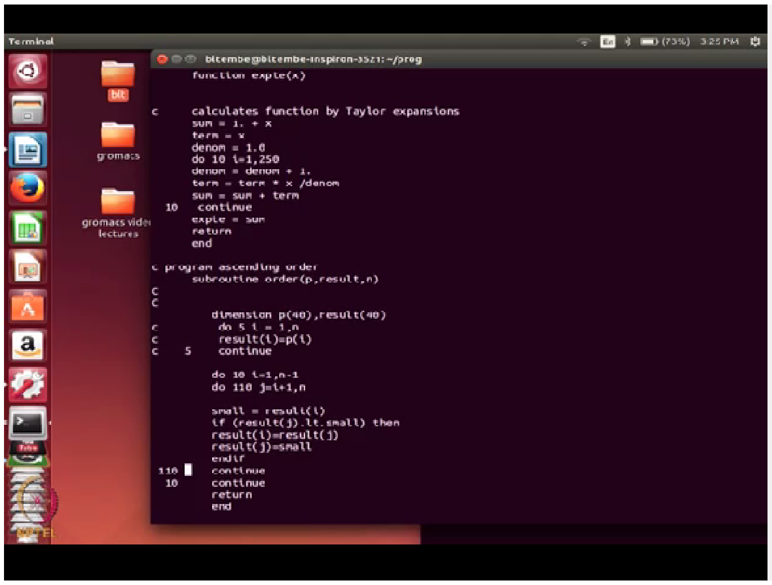
(Refer Slide Time: 29:19)



```
bitembe@bitembe-inspiron-3521: ~/prog
      call order(a,b,n)
      write(*,*) 'the n numbers arranged in ascending order'
      write(*,*) (b(i),i=1,n)
      stop
      end

      function expte(x)


c     calculates function by Taylor expansions
      sum = 1. + x
      term = x
      denom = 1.0
      do 10 i=1,250
      denom = denom + 1.
      term = term * x /denom
      sum = sum + term
   10  continue
      expte = sum
      return
      end

c program ascending order
      subroutine order(p,result,n)
c
c
         dimension p(40),result(40)
c         do 5 i = 1,n
c         result(i)=p(i)
c    5    continue

         do 10 i=1,n-1
         do 110 j=i+1,n

         small = result(i)
         if (result(j).lt.small) then
         result(i)=result(j)
```

And I also make them 40 in the subroutine, okay. So I change in the wrong place, escape, okay. So dimensions are being changed.

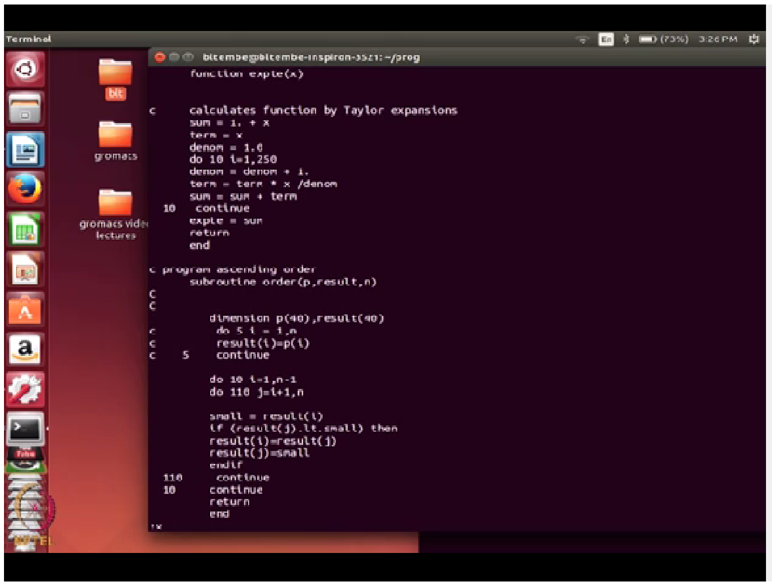
(Refer Slide Time: 29:53)



```
bitembe@bitembe-inspiron-3521: ~/prog
      function exple(x)

c     calculates function by Taylor expansions
      sum = 1. + x
      term = x
      denom = 1.0
      do 10 i=1,250
      denom = denom + 1.
      term = term * x /denom
      sum = sum + term
  10   continue
      exple = sum
      return
      end

c program ascending order
      subroutine order(p,result,n)
c
c
         dimension p(40),result(40)
c         do 5 i = 1,n
c         result(i)=p(i)
c    5    continue

         do 10 i=1,n-1
         do 110 j=i+1,n

         small = result(i)
         if (result(j).lt.small) then
         result(i)=result(j)
         result(j)=small
         endif
 110     continue
  10     continue
         return
         end
```

And when I execute it, my program earlier, see this last line, 100 continue. Instead of, so the position of this is, remember line numbers have to come in the first 5 places. So this is 110, first place is blank, 110, next 3 digits are 110, next is blank, so the next is blank. So instead of giving 110 in the first 5 places, I had given 110 in the, so the 0 was in the sixth place.

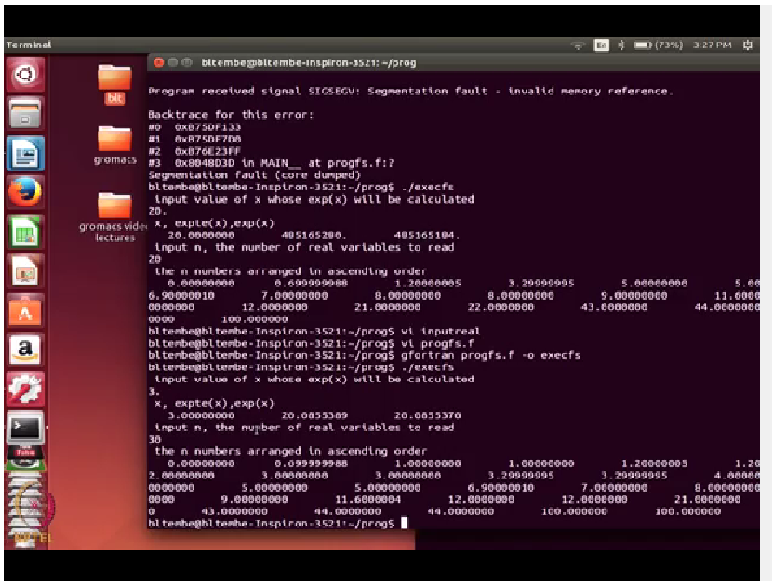
(Refer Slide Time: 30:29)



```
bitembe@bitembe-inspiron-3521: ~/prog
      function exple(x)

c     calculates function by Taylor expansions
      sum = 1. + x
      term = x
      denom = 1.0
      do 10 i=1,250
      denom = denom + 1.
      term = term * x /denom
      sum = sum + term
  10   continue
      exple = sum
      return
      end

c program ascending order
      subroutine order(p,result,n)
c
c
         dimension p(40),result(40)
c         do 5 i = 1,n
c         result(i)=p(i)
c    5    continue

         do 10 i=1,n-1
         do 110 j=i+1,n

         small = result(i)
         if (result(j).lt.small) then
         result(i)=result(j)
         result(j)=small
         endif
  110     continue
  10     continue
         return
         end
:x
```

So what I am saying, instead of 110 here, so now see 1 2 3 4 5 6. So the 0 was in the sixth place and it gave me lots of errors. What is the error I got? It says that end is not found, okay. Do loop is not doing properly. So once you make any mistake somewhere, it can give you error in totally different place. So make sure that such errors are avoided. So now I will compile this.

(Refer Slide Time: 31:09)



```
bltembe@bltembe-inspiron-3521: ~/prog
Program received signal SIGSEGV: Segmentation fault - invalid memory reference.

Backtrace for this error:
#0  0xB75DF133
#1  0xB75DF7D0
#2  0xB76E23FF
#3  0x8048D3D in MAIN__ at progfs.f:?
Segmentation fault (core dumped)
bltembe@bltembe-Inspiron-3521:~/prog$ ./execfs
 input value of x whose exp(x) will be calculated
20.
 x, expte(x),exp(x)
   20.0000000       485165200.       485165184.
 input n, the number of real variables to read
20
 the n numbers arranged in ascending order
   0.00000000      0.699999988       1.20000005       3.29999995       5.00000000       5.00
6.90000010       7.00000000       8.00000000       8.00000000       9.00000000       11.6000
0000000       12.0000000       21.0000000       22.0000000       43.0000000       44.0000000
0000       100.000000
bltembe@bltembe-Inspiron-3521:~/prog$ vi inputreal
bltembe@bltembe-Inspiron-3521:~/prog$ vi progfs.f
bltembe@bltembe-Inspiron-3521:~/prog$ gfortran progfs.f -o execfs
bltembe@bltembe-Inspiron-3521:~/prog$ ./execfs
 input value of x whose exp(x) will be calculated
3.
 x, expte(x),exp(x)
   3.00000000       20.0855389       20.0855370
 input n, the number of real variables to read
30
 the n numbers arranged in ascending order
   0.00000000      0.699999988       1.00000000       1.00000000       1.20000005       1.20
2.00000000       3.00000000       3.00000000       3.29999995       3.29999995       4.0000
0000000       5.00000000       5.00000000       6.90000010       7.00000000       8.00000000
0000       9.00000000       11.6000004       12.0000000       12.0000000       21.0000000
0       43.0000000       44.0000000       44.0000000       100.000000       100.000000
bltembe@bltembe-Inspiron-3521:~/prog$
```

So I am compiling this program, okay. Now I will execute. So I will give 3.0, it calculated the exponential. Now I shall give 30 as my number, 30 I want to give now. When I enter, you will see it has not given any error and it has arranged those 30 numbers in an ascending order. So let me conclude what I have done today. I compiled my programs which had both functions and subroutines.

And while executing, I tried to read more data then what was allowed in the dimension statement and naturally the program cannot accommodate, so it gave an error. So how did we handle that error? We went back to the program, we increased the size of my dimension. When I made dimension 40 in the main program, as well as the subroutine, it executed properly. So we will conclude this lecture today.

So I would urge you to write a matrix multiplication program and include that as a subroutine in your main program and execute it. So try to do it as an exercise. Next time, if necessary, I will just show you the results of that calculation. So do practice every program that we have indicated in our lecture series. So next time, I will continue with interpolation and also the program for interpolation. I will conclude at this point. Thank you.