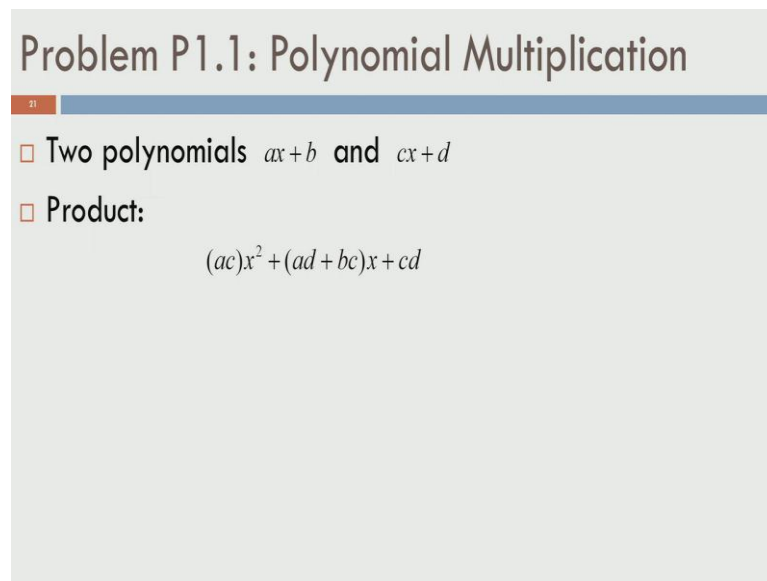


Programming, Data Structures and Algorithms
Prof. Shankar Balachandran
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture – 03
Variables, Assignments and Operators

In module 3 we look at the notion of variables assignments and how operators, in C works.

(Refer Slide Time: 00:25)



Problem P1.1: Polynomial Multiplication

- Two polynomials $ax+b$ and $cx+d$
- Product:
$$(ac)x^2 + (ad+bc)x + cd$$

So, let us like a small problem again, this is a problem on polynomial multiplication, let us say we given two polynomials ax plus b and cx plus d . So, by given what I mean as a given the coefficients a comma b and you are also given the coefficients c comma d and you are expected to compute the product of the polynomials. So, the product could be acx^2 plus ad plus bc times x plus cd . So, what we really want is the coefficient of x^2 , the coefficient of x and cd itself.

(Refer Slide Time: 01:01)

Writing a Program for the P1.1

- Steps
 - ▣ Declare storage for all coefficients
 - ▣ Need to read the coefficients a , b , c and d from the user
 - ▣ Perform arithmetic operations and store the results
 - ▣ Print the coefficients of the resultant polynomial

So, to write a program for this problem P 1.1 we need the following steps. So, we need to declare storage a , b , c , d are all coefficients, if you read from the user you need to store them temporally somewhere. And once the declaration is done you need to actually read the coefficients from the user, perform all the arithmetic operations required and finally, print them on the screen. So, that is the basic steps.

(Refer Slide Time: 01:25)

Close Look at the Program

```
//This is a program to multiply two polynomials  $ax+b$  and  $cx+d$ 
#include <stdio.h>
int main()
{
    int a, b, c, d;
    int p2, p1, p0;
}
```

Allocate storage for a , b , c , d

Allocate storage for p_2 , p_1 , p_0

Datatype: We want the coefficients to be integers

Let us... So, I already have a small program written up here. So, we start with declaration of storage that is step 1. So, again ignore the first few lines, you see that there is `int a, b, c, d;` and `int p2, p1, p0;`. So, this is a very much like what we did earlier. So, we have a , b , c , d so, we need four coefficients and the

product the polynomial has only three coefficients p 2, p 1 and p naught is the name that I am giving them.

(Refer Slide Time: 01:57)

```
Read the Inputs
24
printf("Enter a:");
scanf("%d",&a);
printf("Enter b:");
scanf("%d",&b);
printf("Enter c:");
scanf("%d",&c);
printf("Enter d:");
scanf("%d",&d);
```

Step 2 is read the inputs. So, in this program I written into the such away that will prompt the user for one coefficient at a time. So, there is printf enter a and the user is expected to enter a number and it is read, using the statement scanf, then you have printf enter b now you are ready to read d and so, on. So, four coefficients are read so, this step 2.

(Refer Slide Time: 02:25)

```
Calculate Coefficients and Print
25
p2 = a*c;
p1 = a*d + b*c;
p0 = b*d;

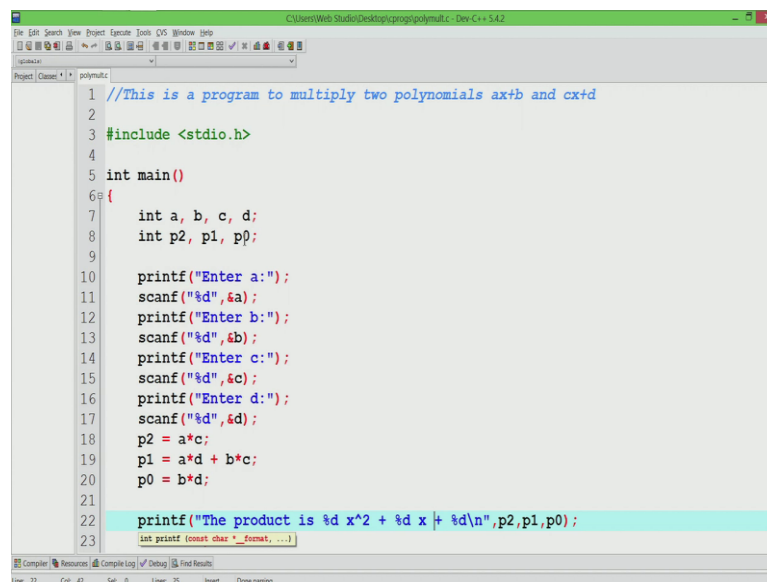
printf("The product is: %dx^2+%dx+%d\n",p2,p1,p0);
return 0;
}
```

Step 3, calculate the coefficients and print. So, here there are three lines p 2, p 1 and p naught, p 2 is a time c, p 1 is a into d plus b into c and p naught is b into d. So, this the very first time we are seeing an assignment. So, the way to look at assignments is look at the equality sign in the first line here, you see a star c. So, star is the operator for multiplication. So, you are multiplying a with c and that is assigned to p 2.

So, the way to look at this line is you do the operation a time c and the values assigned to p 2 which is on the left side, then you have a times d plus b times c. So, you do this operation and you assign it to p 1 and b into d is assigned to p naught. So, at the end of these three statements, we expect p 2, p 1 and p naught to reflect the point that you have already taken care of the multiplication of the polynomial. And now we are ready to print and the product is printed on the screen.

So, again forget all the fancy looking thinks there. So, you can see the printf the product is and something there. So, ignore it for now, it is only printing the coefficients along with the... in the polynomial format itself. So, let us run this program I have written this program up and let us run this program.

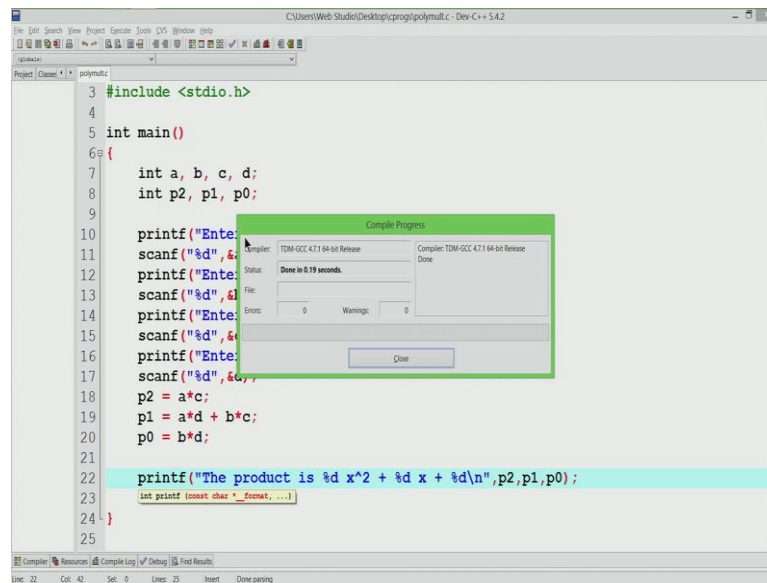
(Refer Slide Time: 03:56)



```
1 //This is a program to multiply two polynomials ax+b and cx+d
2
3 #include <stdio.h>
4
5 int main()
6 {
7     int a, b, c, d;
8     int p2, p1, p0;
9
10    printf("Enter a:");
11    scanf("%d", &a);
12    printf("Enter b:");
13    scanf("%d", &b);
14    printf("Enter c:");
15    scanf("%d", &c);
16    printf("Enter d:");
17    scanf("%d", &d);
18    p2 = a*c;
19    p1 = a*d + b*c;
20    p0 = b*d;
21
22    printf("The product is %d x^2 + %d x + %d\n", p2, p1, p0);
23    list_printf(const char * _format, ...)
```

So, you can see that the program is same as what I said. So, you have int a comma b comma c comma d, then you have the three coefficients p 2, p 1 and p naught. The four scan statement for reading the variables, three statements for doing all the assignments or doing the arithmetic operations and one final statement which takes care of printing.

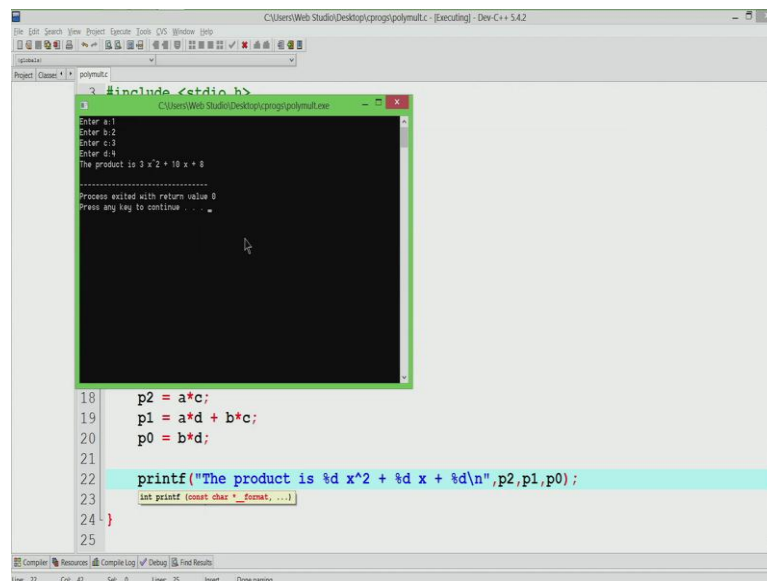
(Refer Slide Time: 04:22)



```
3 #include <stdio.h>
4
5 int main()
6 {
7     int a, b, c, d;
8     int p2, p1, p0;
9
10    printf("Enter a: ");
11    scanf("%d", &a);
12    printf("Enter b: ");
13    scanf("%d", &b);
14    printf("Enter c: ");
15    scanf("%d", &c);
16    printf("Enter d: ");
17    scanf("%d", &d);
18    p2 = a*c;
19    p1 = a*d + b*c;
20    p0 = b*d;
21
22    printf("The product is %d x^2 + %d x + %d\n", p2, p1, p0);
23    int printf(const char * _format, ...);
24 }
25
```

So, I will compile it now, it gives me no errors or no warnings,. So, far I am shown you what errors and warnings could be I written programs which are all correct. Now, I am ready to run.

(Refer Slide Time: 04:32)



```
18    p2 = a*c;
19    p1 = a*d + b*c;
20    p0 = b*d;
21
22    printf("The product is %d x^2 + %d x + %d\n", p2, p1, p0);
23    int printf(const char * _format, ...);
24 }
25
```

So, as before we see the value prompt for a. So, I am going to enter the polynomial 1 comma 2. So, essentially 1 x plus 2 for the first polynomial and 3 x plus 4 for the second polynomial. So, the result is 3 times x squared plus 10 x plus 8. So, it is 1 x plus 2 times 3 x plus 4. So, that is 3 x squared 10 x plus 8 ((Refer Time: 05:08)). So, let us look at the program in it is entirety, we have the first few lines which has the comment.

So, there is a comment on the top here which says, this is the program to multiply two polynomials $ax + b$ $cx + d$. So, it is always good to show the intention of the programmer and this kind of documentation is helpful for somebody else who is going to read your program and understand what is happening. Then, as before we have this line `hash include stdio dot h`, this is necessary for doing all the IO operations.

If you want to read from the keyboard or if you want print on the screen, you will need this and then we have the declarations for `a`, `b`, `c`, `d`, `p 2`, `p 1` and `p naught`. Then, we did a series of printing on the screen and reading from the user and we did the operations and printed the result on the screen. So, I already showed you the execution. So, this by itself is not what I want to show, I want to drive something else, let us take a close look at the program and see some of the nitty gritty details that are involved.

So, let us start with `a comma b comma c comma d`. So, I assumed in the polynomial is $ax + b$ and $cx + d$. So, however, we need the user to give these four coefficients and these four coefficients have to be taken from the user and stored temporarily somewhere and that is where we are going to use these variables `a`, `b`, `c` and `d`. So, in a little while explain what the word variable means, at this point just assume that `a`, `b`, `c` and `d` are four variables.

And it is not enough to just have these coefficients, because you are supposed to produce the product terms. So, you have `a`, `b`, `c`, `d` and you have `p 2`, `p 1`, `p naught` and when you have `int a comma b comma c comma d`, it actually allocates storage for `a`, `b`, `c` and `d`. Because, once you read it from the user, you have to save it somewhere and when you have `p 2`, `p 1`, `p naught` again you allocate this storage for `p 2`, `p 1`, `p naught` and store it somewhere.

So, essentially it takes care of all the storage that you need to do all the operations and this storage is going to be in the main memory. So, we already saw the model for the computer. So, we have memory and we have a CPU and we have a other units. So, these variables will be stored in the memory and if you look at this `int`, `int` basically means that `int` stands for integer and it means that anything that follows this is going to be a variable which is of an integer type.

So, when you have `int a comma b comma c comma d`, we have four integers or four variables of the data type called integers. Similarly, we have `p 2`, `p 1`, `p naught` which also have this `int` before it, which means there are three integers by name `p 2`, `p 1` and `p`

naught. So, if the input coefficients a, b, c, d are all integers, the product and sum of integers are always integers. So, therefore, I have used integer for p2, p1, p naught also and this is called the data types.

So, whenever you want a variable it cannot be a variable of an unknown type, it has to be a variable of a known type. And there are a few basic data types that the language provides you, there are other data types that you can build on your own, in this case I am going to use basic data type provided by C which is called int. So, remember int stands for integer.

(Refer Slide Time: 08:56)

The slide is titled "From a Memory Point of View". It contains the following text and diagram:

- Initially unused
- `int a, b, c, d;`
- `int p2, p1, p0;`

On the right side, there is a vertical stack of seven empty rectangular boxes representing memory locations. The boxes are labeled from top to bottom as: a, b, c, d, p2, p1, and p0. A mouse cursor is pointing at the 'a' box.

Now, let us see what is going to really happen from the memory point of view. So, let us see the program. So, initially the memory is unused. I am showing a segment of the memory here. So, let us assume that the memory is huge and it has several bytes of storage. I am showing only a portion of the memory on the right-hand side. So, initially everything is unused and the movement there is a declaration `int a, b, c, d;`. What it really means is, when you run the program, there are going to be four memory locations which will be called by the names a, b, c, and d. So, the memory that you have is initially uncommitted, the declaration `int a, b, c, d;` will attach some names to the memory locations. So, in this example, this memory location is going to be called a, this memory location is called b, this is called c, and this location is called d.

So, I am purposely showing these locations without any values. Because, when you declare, you should not assume that there is going to be some known value already.

present in the memory. So, when you start the program we do not know what the values are and we also expect the user to input these values a, b, c and d. And the next line was `int p2, p1, pnaught`, which means there are going to be three memory locations which are going to be called p2, p1, pnaught.

So, in this picture even though it shows a, b, c and d to be continuous and p2, p1, pnaught also to be continuous, it does not necessarily have to be soon. So, they do not have to be continuous memory locations. So, we have this. So, we look at the first few lines of the main function and all it does is, it has declaring storage. So, you are saying various memory locations are going to be called by these names.

(Refer Slide Time: 11:00)

Reading Inputs: How it changes memory

```
printf("Enter a:");
scanf("%d",&a);
printf("Enter b:");
scanf("%d",&b);
printf("Enter c:");
scanf("%d",&c);
printf("Enter d:");
scanf("%d",&d);
```

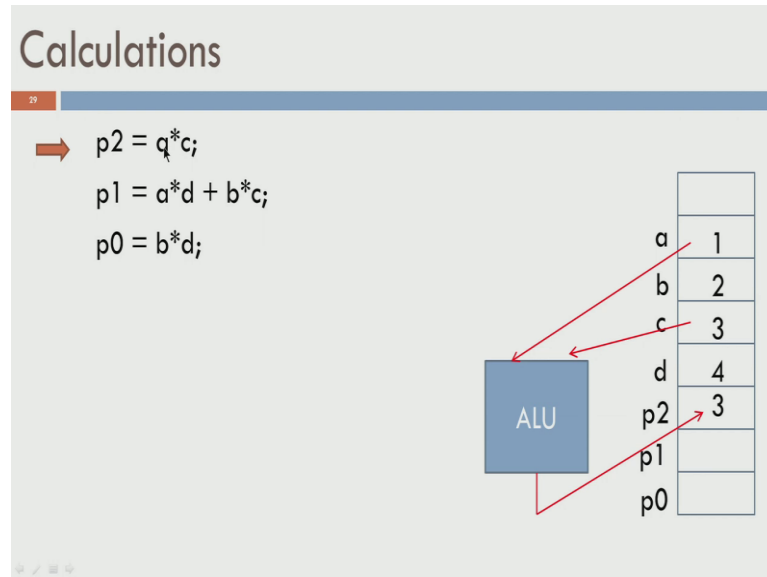
a	1
b	2
c	3
d	4
p2	
p1	
p0	

Now, let us look at the statements that follow. So, we have `printf enter a` and let us say it prints this enter a on the screen, it is prompting the user and when it is comes to `scanf`, it is going to read one number from the screen, let us assume that the user actually input 1. So, we assume that the polynomial is ax plus b is $1x$ plus 2 and cx plus d is $3x$ plus 4. So, at this point the user is expected to have entered 1 and that would go into the memory location which is called a.

So, one goes into this memory location, then `printf enter b` that I will show enter b on the screen, when the user inputs b in this case it is 2 that would go into this memory location and so, on and so, forth. So, at the end of all the `scanf` statements shown here, if the user entered 1, 2, 3, 4 these locations would contain 1, 2, 3, 4 p2, p1, pnaught I am showing blank, it does not mean that it does not have any value, it only means that we do not

know the value yet it has to be calculated yet. So, at this point a, b, c, d are all ready and is available in the program.

(Refer Slide Time: 12:20)



So, let us move to the next set of statements, the next set of statements are for calculation of p 2, p 1 and p naught. So, remember in this computer's model, I showed control unit the CPU which contains the control unit and the ALU you had memory. So, in this at this point I am showing the memory and the ALU, I am not showing the input and output and other notes and so, on. So, ALU stands for arithmetic and logic unit and since we need arithmetic operations here, like multiplication, addition and so, on I am also showing the ALU.

So, let us see the very first statement here p 2 is a time c. So, this star means multiplication. So, what you want is a and c to be multiplied and the result of that must be stored in p 2, this is what you want, and the way it works is this. So, when you have p 2 is a time c, a and c are brought in from the memory into the ALU. The ALU does the operation multiplication, in this case it's 1 times 3 and the result 1 into 3 which is 3 is written back into p 2.

So, if you look at this statement you read the value of a, which is 1, you read the value of c which is 3, you do multiplication the result is 3 and the result is written back in the memory location p 2, which is what we have. So, far, we read 1 and 3 in the result is 3 here. So, now, we are ready to move to the next statement, the next statement is slightly more complicated, it has a star, it has a plus and it has another star. And the order in

which the operation is done is, you perform the operation $a \times b$ first, you then perform the operation $b \times c$, the result of these two or then added up and the end result is put back into p_1 .

So, that is what we are going to see the animation now. So, a and d are both read into the ALU and it is not ready to be written back to p_1 yet, because you still have more operations on the right hand side. So, this 1×4 the result is 4, it is temporarily stored inside the ALU itself. Then, you have $b \times c$, b and c are read from the memory and the result is 6. So, 2×3 is 6 and you cannot right 6 to p_1 yet, because this is also a temporary result, what we really want is $a \times d$ plus $b \times c$. So, we cannot write 6 to the memory yet.

At this point the ALU has 4 and 6, which are required for this addition operation to be performed. So, if you have only one of these operation let say $a \times d$ was available, but $b \times c$ was not available, then the addition cannot be performed. But, at this point of time you have both 4 and 6 and $4 + 6$ is executed by the ALU the result is 10. And once you have the result, you're ready to write it back into p_1 .

So, the RHS or the Right Hand Side is ready, we have the result and we write the result into p_1 next the CPU moves to the last line, which is p_0 is $b \times d$. So, again b and d are brought from the main memory this time. So, b and d changed they are 2 and 4 and the result of that is written back into the main memory. So, at the end of the program we have 3, 10 and 8. So, let us take a quick look at the program here ((Refer Time: 16:13)).

So, what we saw was we these four or these statement from line number 10 to 17. Initially filled up the four memory locations a , b , c and d with values by reading from the user, lines 18, 19 and 20 did operations the user is not involved here, the values from the memory are read and the calculations are performed, the results are written back into the memory. And finally, when line number 22 is executed, it is asking p_2 , p_1 and p_0 to be printed on the screen, in a certain format and the way it is going to be printed is as follows.

So, you have the product is percentage $d \times d$ plus percentage $d \times d$ plus percentage d back slash n what that means, is. So, if you look at this percentage d and if you run this program, you will never see this percentage d on the screen. So, I enter the same thing if you see the result, the product is only $3 \times d$ plus $10 \times d$ plus 8. What is really doing

is, this percentage d is called a format specifier. So, where ever you have percentage d here, you look at what is the value outside the codes.

So, it says print p 2 as integer that is what this percentage d means. So, p 2 is an integer, print it as an integer, p 1 is also an integer, print it as an integer also and p naught s an integer, print it as an integer also. So, that is why you get this result 3 times x squared plus 10 x plus 8, this x char at 2 or x squared is verbatim in the program itself and this x plus is also verbatim in the program, you see that in the result here and finally, 8.

So, we will get into the format specifier and so, on in more detail later. But, as of now just remember that this percentage d is not something that is printed on the screen. So, to print something on the screen, you need to p 2, p 1, p naught stored in the main memory, but we have them as 3, 10 and 8. So, when you see the printf statement you can read these 3, 10 and 8 and print them on the screen.