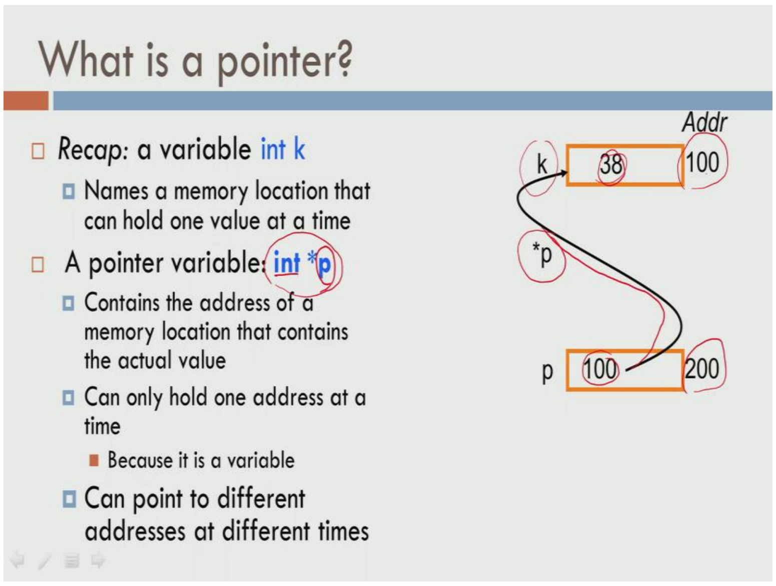
Programming, Data Structures and Algorithms

Prof. Shankar Balachandran
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Module - 9B
Lecture – 14
What is a pointer?
Pointer variables, their declaration and their contents
Example of pointers and illustration on IDE
Declaring pointer does not allocate memory

In this module, what we are going to look at is very interesting aspect that C and several other programming languages give you, which is not just accessing values that are stored in variables, but also getting access to the memory locations or the addresses.

(Refer Slide Time: 00:34)



What is a pointer?

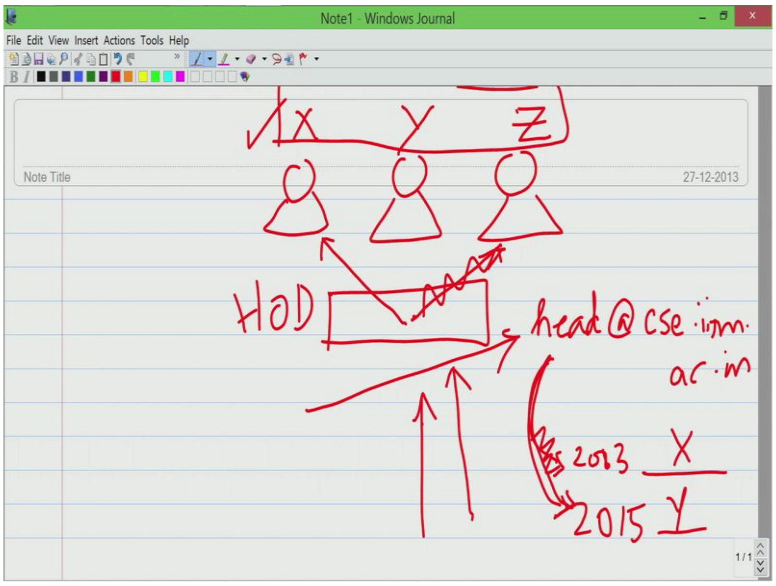
- Recap: a variable int k
  - Names a memory location that can hold one value at a time
- A pointer variable: int *p
  - Contains the address of a memory location that contains the actual value
  - Can only hold one address at a time
    - Because it is a variable
  - Can point to different addresses at different times

So, what is a pointer? So, to do that let us remember what happens, when we declare a variable called k. So, if I do int k, k is the name of a memory location that can hold a single value at a time. So, for instance if I have int k, k could be a given array location 100 and at some point of time, you could have k equals 38 and the value 38 will get into this location called k. So, from the programs perspective, we are going to use k which is the name of the variable, internally it is stored in location 100 and the value that is stored is 38.

So, that is the concept of a variable. Now, what is a pointer? A pointer is essentially a variable and in this example, we have this thing called int star p. So, the key thing to note here is this thing called star, it is not int p, it is int star p. So, remember star is not a valid character in a variable name for C. So, this star is reserved for something specific. In this case, p is supposed to be a pointer to an integer. So, let us see what this does.

If I declare something called int star p, this p can contain memory locations instead of the actual value. So, let us see what that means, so p is a variable as we had before which means, your compiler is going to allocate some location for it. So, let us say p as a variable is allocated location 200 and when you say int star p, it means it is a pointer. If I put int star p equals to 100, the value 100 gets into this location 200 and if I do star p, star p is an operation that you are doing. The meaning of that is, you take location 100 see, what that is pointing to. So, I take the value 100, instead of reading it as a value I read that as an address and in that address, what is the value that is present? So, the value that is present is 38, so p is an integer pointer. What that means is, the value that is stored that you are going to reference to is actually an integer, p is just pointing to it, p does not contain the value 38, but p contains the address in which 38 is stored. So, one way to look at this is as follows.

(Refer Slide Time: 03:35)



So, let us say we have, so in our department we have a HOD and so the Head of the Department. So, it is actually a person, so let us assume that there are people and there are several people who could be HOD’s. But, at any point of time, there is only a single person who is the head of the department. Let us say, you as a student walk in and you want to find out, who the HOD is. So, of course you could go and ask each one of these faculty members who the HOD is. But, the notion of HOD is always just one person.

Let us say I have, this small marker which says, the current HOD is person professor Z. So, let us assume that there are three professors, X, Y and Z and you come in and ask who is the current HOD and I say that the current HOD is Z. So, that way from outside, you do not have to know who the current HOD is. You ask, who the current HOD is and you get pointed to professor Z. At some point of time in the future, professor Z steps down as HOD and let us say professor X takes of the headship, then the pointer points to X.

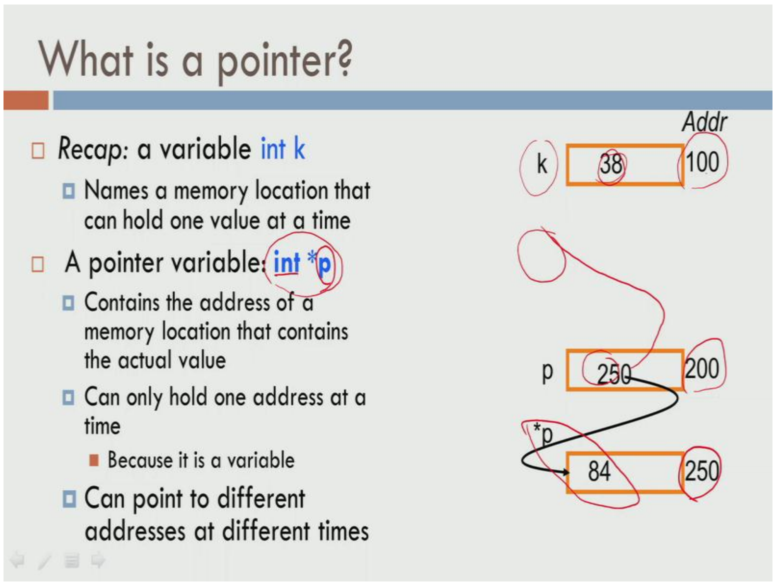
So, again let us say somebody else walks in and ask the same question, who is the current HOD? This pointer, can now point to X and not Z. So, that way what you are actually getting is not the actual value, which is X, Y or Z, but you are getting the pointer to the actual value. So, this is very useful for instance, this is something that we have in our department. So, in our department we have an email address called head@cse.iitm.ac.in.

So, this head@cse.iitm.ac.in is a permanent email address, but this in turn points to different people at different times. So, for instance our current HOD in year 2013, his email address is pointed to now. When his turn comes to give up the position, let us say in 2015 somebody else becomes the HOD, let us say professor Y becomes the HOD. Then, I can remove the pointer pointing to professor X and I can instead make that pointer point to professor Y and this way, you always will email head at CSE and the internal pointer will forward it to Professor X or Y or Z, according to who the current HOD is.

So, this is a very useful concept and it can come in very, very handy, we will see examples of this later. But, as of now just remember that p is just like any other variable. It gets a memory location, but the value in that is not a integer or a floating point value or a character, it is going to be a memory address. And this memory address, you can think of it as pointing to some location and when you do star p, you are looking at the address and what it is pointing to and from there, what is the value that you get.

So, just like any other variable p can hold only one address at a time, but you can point to different addresses at different times. So, just like variables can have different values at different times, a pointer variable can have different addresses at different points of times.

(Refer Slide Time: 07:23)



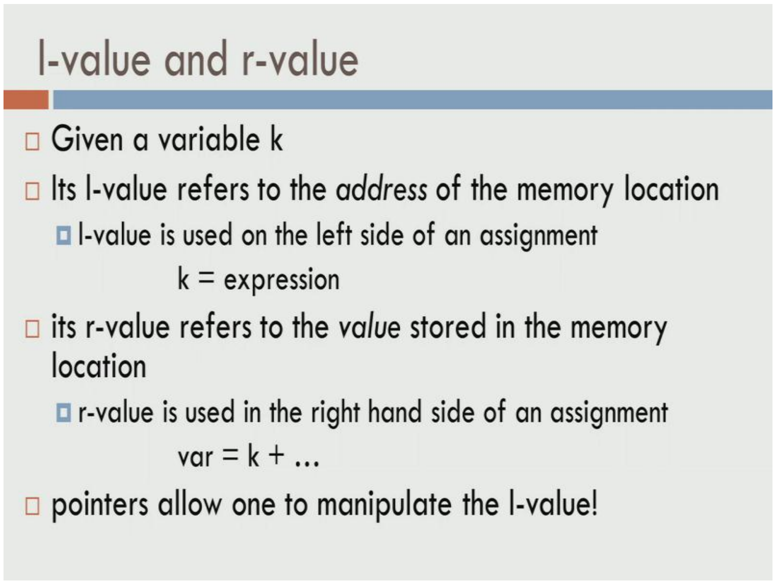
What is a pointer?

- Recap: a variable int k
  - Names a memory location that can hold one value at a time
- A pointer variable: int *p
  - Contains the address of a memory location that contains the actual value
  - Can only hold one address at a time
    - Because it is a variable
  - Can point to different addresses at different times

Addr
k
38
100
p
250
200
*p
84
250

So, let us say I change the pointer value from 100 to 250. I change p from taking the value 100 to 250, then it can point to a new location 250 and star p would now be 84 and not 38.

(Refer Slide Time: 07:43)



l-value and r-value

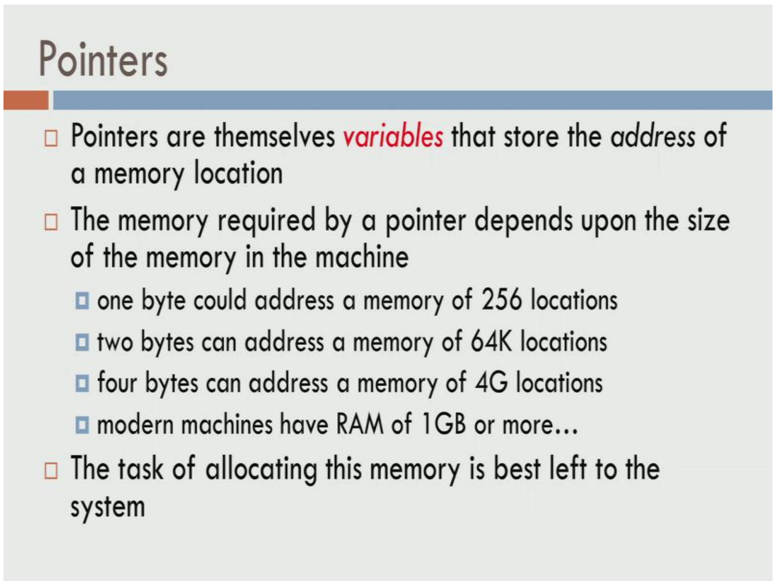
- Given a variable k
- Its l-value refers to the *address* of the memory location
  - l-value is used on the left side of an assignment

    k = expression
- its r-value refers to the *value* stored in the memory location
  - r-value is used in the right hand side of an assignment

    var = k + ...
- pointers allow one to manipulate the l-value!

So, to understand this, it may be useful to know this concept of what is called l value and r value. Let us say, I am given a variable k, the l value refers to the address of the memory location. So, l value is used on the left side, so whenever I say k, in turn it is actually a specific memory location, that is the l value and the r value is the actual value that you are going to put in. So, if I say k equals x plus y, then x plus y is going to be a value and the value of this is going to be stored in k. So, the value of x plus y is the r value and the address of k is the l value. So, pointers are actually allowing you to manipulate the l value. You always have expressions on the right side, you get the r value always, but pointers are mechanism by which you actually can get access to the l value also.

(Refer Slide Time: 08:43)

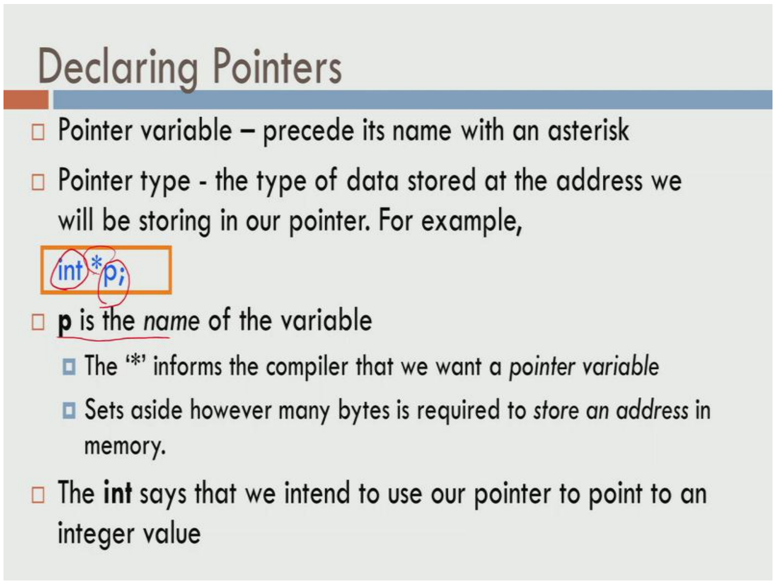


Pointers

- Pointers are themselves *variables* that store the *address* of a memory location
- The memory required by a pointer depends upon the size of the memory in the machine
  - one byte could address a memory of 256 locations
  - two bytes can address a memory of 64K locations
  - four bytes can address a memory of 4G locations
  - modern machines have RAM of 1GB or more…
- The task of allocating this memory is best left to the system

So, pointers are themselves variables that store the address of the memory location and the memory required by a pointer depends upon the size of the memory in the machine. So, for instance let us say I have a pointer of size 8 bits or 1 byte, it can only point to 256 different locations. But, if I have a pointer of 2 bytes in size, it can point to 64000 or 64K locations, so that is 65536 locations. If it is four bytes, it can point to 4 gigabyte locations or 2 power 32 locations and so on.

(Refer Slide Time: 09:22)



Declaring Pointers

- Pointer variable – precede its name with an asterisk
- Pointer type - the type of data stored at the address we will be storing in our pointer. For example,

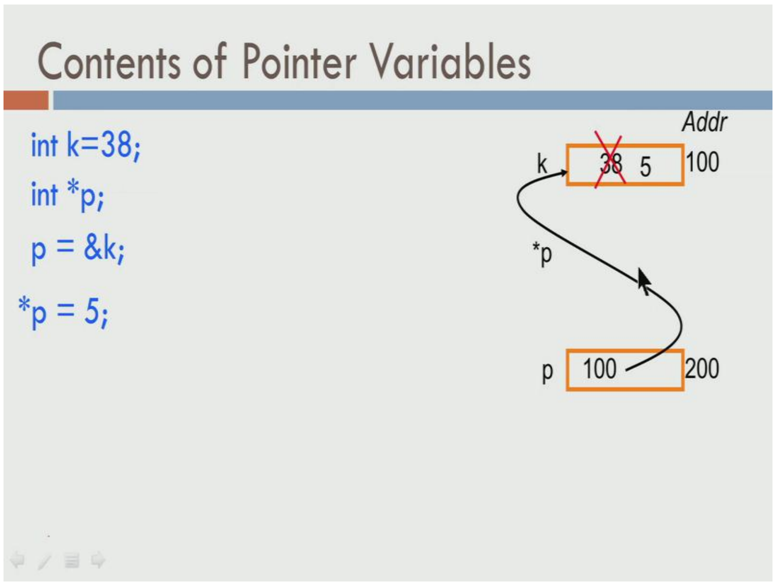
```
int *p;
```

- **p** is the *name* of the variable
  - The '*' informs the compiler that we want a *pointer variable*
  - Sets aside however many bytes is required to *store an address* in memory.
- The **int** says that we intend to use our pointer to point to an integer value

So, let us see how to declare a pointer variable. You precede the name, so p is the variable name and you precede a variable name by this mark called asterisk or a star right and before that you have the data type. So, there are three things that go with the pointer variable, you have the data type, you have star and you have the variable name.

So, p is the name of the variable and the star symbol there, informs the complier that we want a pointer variable, it should not be an integer p, it should be a pointer p which means, you are going to store addresses in them and not integer values and the compiler will set aside, how many ever bytes you want to store the address. And this thing int before that says, we intend to use this pointer to point to integer values. So, you could have floating point values and so on, in which case you need a floating point pointer.

(Refer Slide Time: 10:28)



Contents of Pointer Variables

```
int k=38;
int *p;
p = &k;
*p = 5;
```

So, let us see what the contents of the pointer variables are. So, there is a small piece of code here on the left side. So, int k equals 38 which means, there is a variable called k, whose values is going to be 38 and you have int star p and this int star p, so I have int k equals 38 which means I have an integer variable called k, int star p is a pointer variable called p and then we are going to do some operations later. So, let us see what this means.

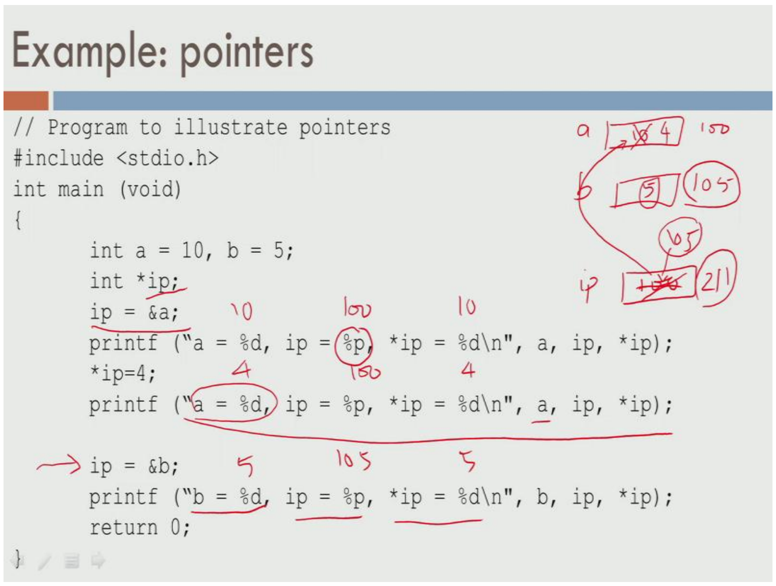
So, int k equals 38, what does that do? The compiler sees that k is an integer of data type integer. So, k is a variable of data type integer, so it gives you a memory location. In this case, I am assuming that it is 100 from the program side, we are going to call it k and the value that is stored is 38. So, this is something that you already know. Now, let us look at the next line int star p. So, p is a variable name, so every variable gets a memory location, so p gets a memory location. In this case, I am assuming that it is 200, but what type is p, p is of the type int star or it is a pointer to integer. So, int star p is a pointer to integer as of now, I do not have anything to initialize here, so initially it is a unknown value for p, next statement says p equals ampersand of k. So, the meaning of ampersand of k is give the address of k, not the contents of k, give the address of k and what is the address of k, it is 100.

So, when you say p equals ampersand k, p gets the value 100. Remember, it is not 38 it gets 100 at this point, you can imagine that p is pointing to the address of k and when you say star p equals 5, it means take p which is at location 200. It is supposed to be a pointer which is 100 and star p is the contents pointed to by 100, which is now 38. It says change it to 5, so star p is 38, change it to 5, so let us do this once more.

So, p is at location 200, when you say star p equals 5, you first go to location 200 and find out, what is the value there. The value is 100 and star p means the value that is pointed 2 by p, which is currently 38. So, you take 38 and you want star p to be 5 which means, this 38 should be change to 5. So, it is not changing this 100 to 5, it changing this 38 to 5. So, if the pointer is still pointing to location at which k is available.

So, at this point if we go and print k, k will actually have the value 5 and not 38, because k is address 100 and you used p to manipulate the value at address 100.

(Refer Slide Time: 13:58)



Example: pointers

```
// Program to illustrate pointers
#include <stdio.h>
int main (void)
{
        int a = 10, b = 5;
        int *ip;
        ip = &a;
        printf ("a = %d, ip = %p, *ip = %d\n", a, ip, *ip);
        *ip=4;
        printf ("a = %d, ip = %p, *ip = %d\n", a, ip, *ip);

        ip = &b;
        printf ("b = %d, ip = %p, *ip = %d\n", b, ip, *ip);
        return 0;
}
```

So, I have a small program here which shows this in more detail. So, there are two integers, a and b, a currently holds the value 10 and b currently holds the value 5 and this int star ip is a pointer. So, ip is an integer pointer and currently it is uninitialized, so at this point you can imagine that a and b are two locations, so I am going to do a small diagram on the right side. So, a is some location, let us assume that it is 100 and a equals 10, we will put 10 at location 100, b is a another variable let us say this is at location 105 and it is going to take a value 5 and ip is a pointer variable.

Let us say, this is at location 211 and since there is no initialization, there is unknown value here, it is unknown. Let us look at the first statement here, ip equals ampersand of

a. So, ampersand of a is the address of a, so what is the address of a? It is 100. So, this will contain the value 100, then let us look at the printf statement here. So, it says print a print, print ip and print star ip. These are three things, we are going to print.

So, what would print a give us? It will give us the value 10 itself, because all the variables when you access them, you directly get the value and what you get for ip. So, here we are also printing ip. What would be the value of ip? Ip should give you 100, because 100 is the address stored in ip and star ip is the contents of the address that is stored in ip, that is the way to interpret it. Star ip is give me the contents of the address that is stored in ip.

So, the address that is stored in ip is 100, the contents of 100 is 10. So, this printf statement is expected to print 10. It is expected to print 10 here, it is expected to print 100 and it is expected to print 10 here, as well. Then, let us look at this line, the next line says i star ip equals. So, star ip equals 4, let us see what that does, so star ip, what is star ip? So, it says change the contents of the address that is pointed by ip to 4.

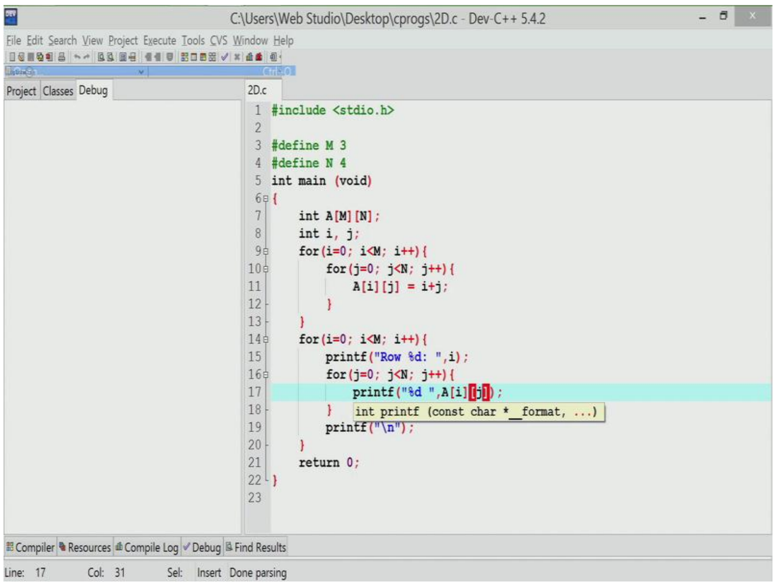
So, ip is still pointing to 100, the contents of that should be is actually 10 now, it says change that to 4. So, contents of the integer pointer pointed by ip should be changed to 4. So, you go here, you look at 100, 100 is pointing to a now. So, 100 is pointing to a and you want to change that to 4. So, this printf statement, so since you have changed that to 4, you would expect star ip to be 4, but what happens because of that is, since a is in location 100, this a equals percentage d.

This is supposed to print the value of a, this will now be 4, ip is still address 100 you want 100 there, ip has not changed. So, this is still be 100 and star ip is now take the address contained in ip, chase that pointer find out the value that pointer is 4. So, this will also print 4, so star ip and a contain the same value, so, so far so good. Now, let us see what happens, because of this statement. We have ip equals ampersand of b, we are now changing the value of ip itself.

So, ip still resides at location 211 and you have ip equals ampersand of b. So, instead of 100 now you will have ampersand of b that is 105. So, this will be 105, so you over wrote 100 with 105. Now, if you take a step back and think about, what star ip must be? So, let us look at star ip, star ip is take the contents of p which is 105 and find out the value that is stored in location 105 that should be 5. So, b is already 5, ip will now have 105 and star ip is the contents of the address 105 that is 5.

So, one thing that is new in this program is, we have used this format specifier called percentage p instead of percentage d. So, percentage d is meant for integers and percentage p is meant for printing the pointer data types.

(Refer Slide Time: 19:51)



C:\Users\Web Studio\Desktop\cprogs\2D.c - Dev-C++ 5.4.2

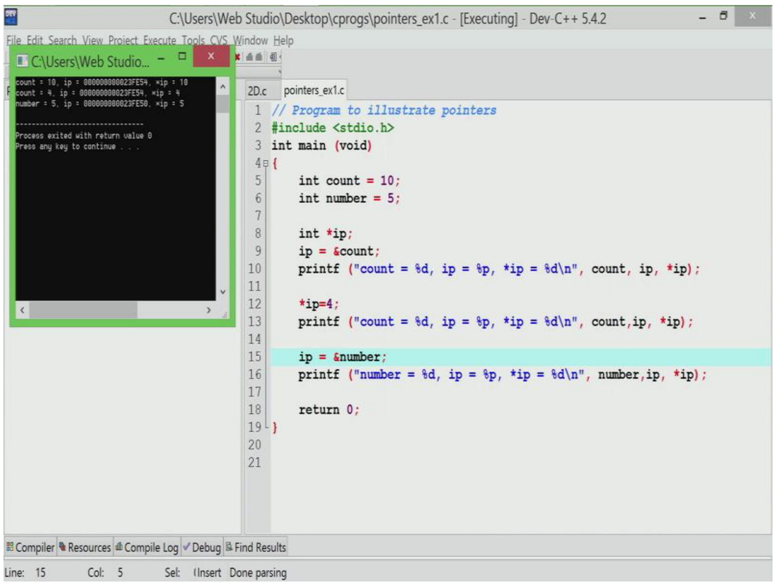
```c
#include <stdio.h>

#define M 3
#define N 4
int main (void)
{
    int A[M][N];
    int i, j;
    for(i=0; i<M; i++){
        for(j=0; j<N; j++){
            A[i][j] = i+j;
        }
    }
    for(i=0; i<M; i++){
        printf("Row %d: ",i);
        for(j=0; j<N; j++){
            printf("%d ",A[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

```
int printf (const char *__format, ...)
```

Line: 17 Col: 31 Sel: Insert Done parsing

So, let us see a small program which is essentially going to run this thing. So, I am going to look at pointers exercise.

(Refer Slide Time: 20:03)



C:\Users\Web Studio\Desktop\cprogs\pointers_ex1.c - [Executing] - Dev-C++ 5.4.2

```
count = 10, ip = 000000000023FE54, *ip = 10
count = 4, ip = 000000000023FE54, *ip = 4
number = 5, ip = 000000000023FE50, *ip = 5

--------------------------------
Process exited with return value 0
Press any key to continue . . .
```

```c
// Program to illustrate pointers
#include <stdio.h>
int main (void)
{
    int count = 10;
    int number = 5;

    int *ip;
    ip = &count;
    printf ("count = %d, ip = %p, *ip = %d\n", count, ip, *ip);

    *ip=4;
    printf ("count = %d, ip = %p, *ip = %d\n", count,ip, *ip);

    ip = &number;
    printf ("number = %d, ip = %p, *ip = %d\n", number,ip, *ip);

    return 0;
}
```

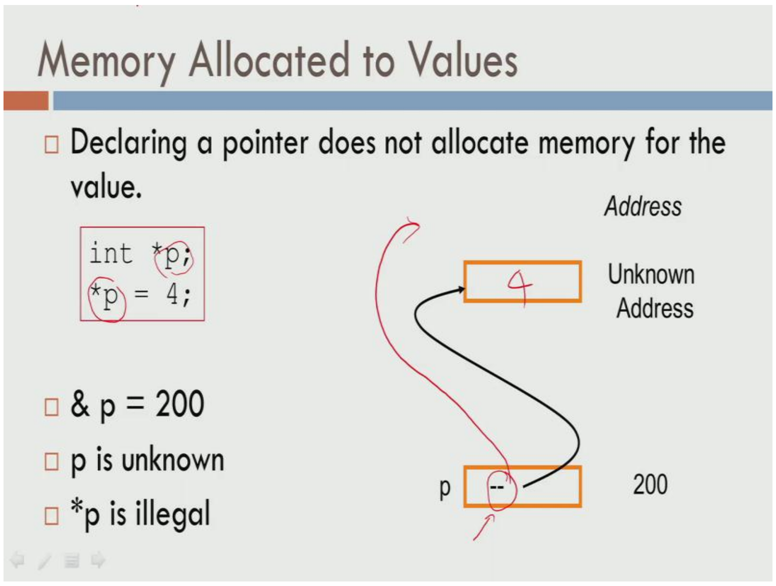
Line: 15 Col: 5 Sel: Insert Done parsing

So, I have the same program here, so instead of a and b, it is called count and number here. So, count equals 10 and number equals 5 and you have these things here. So, let us compile it and run it. So, I want you to pay attention to what is happening here. So, initially count equals 10, so look at this program here, so let us look at this line number. So, this line is saying print count, print ip and print star ip. So, count is 10, ip is some zeros, followed by 2, 3, fe, 5, 4.

So, this is actually an address, do not worry about why it is printing f and so on. So, it is printing something in hexadecimal format. So, that is the address of count and star ip is 10, as we expected. So, because ip is ampersand of count, then we have star ip equals 4, which means change the value of the address that you already have, change the contents to 4. So, in the second line we can see, count is also 4, ip is the same address and star ip is 4 and in line number 15, we changed ip to point to number instead and numbers address is some other address, it is 2, 3, fe, 5, 0 and not 5, 4.

So, you can see that ip changed to 2, 3, fe, 5, 0. The value is 5 and through star ip, you actually get the same value 5. So, instead of a and b this program has count and number. So, hopefully this explanation and the program there, helps you in understanding what is happening.

(Refer Slide Time: 22:09)



Memory Allocated to Values

- Declaring a pointer does not allocate memory for the value.

```
int *p;
*p = 4;
```

- & p = 200
- p is unknown
- *p is illegal

So, one minor point and this is something that many novice programmers in C mess up is the actual notion of memory allocated to values. So, declaring a pointer does not allocate memory for the value, so let me explain what I mean by that. Let us say, I have int star p, so I have a declaration called int star p. What this is supposed to do is, I am going to have a variable called p, which is of the data type integer pointer. Let us say that is at location 200. Since, I have no initialization I have no known value of p. So, that is what you see here.

So, here you see that there is a dash dash, there is no initialization and let us say, I do this star p equals 4. What is this supposed to do? This is an hither to unknown value and this star ip is saying take the address stored in p, chase that pointer and initializes the value to or change the value to 4. But, since this is unknown, you are saying take this unknown, go to some memory location I do not know which and change that to 4.

So, this is pointing to some unknown address and you want this unknown address to have the value 4, this is not a legal operation in C. So, at some point you have to take p and put a valid address in it and only then, you will be able to access star p. So, let us see what the sequence of things are, ampersand of p is 200, because p is given a memory location, compiler will do that because it is a variable, you will get location 200.

However p itself is unknown, so we access the variable p, you get the contents of this location 200. This is unknown and therefore, star p is illegal. So, this brings us to the end of module 2 and in module 3 and 4, we will look at various other subtleties related to pointers. We will not only see pointers as a problem of it is own, but we will also see, how to use pointers and access arrays and what are the different issues that come up, when we use pointer to access arrays. So, you have reached the end of the module 2, we will see these things in modules 3 and 4.

Thank you.