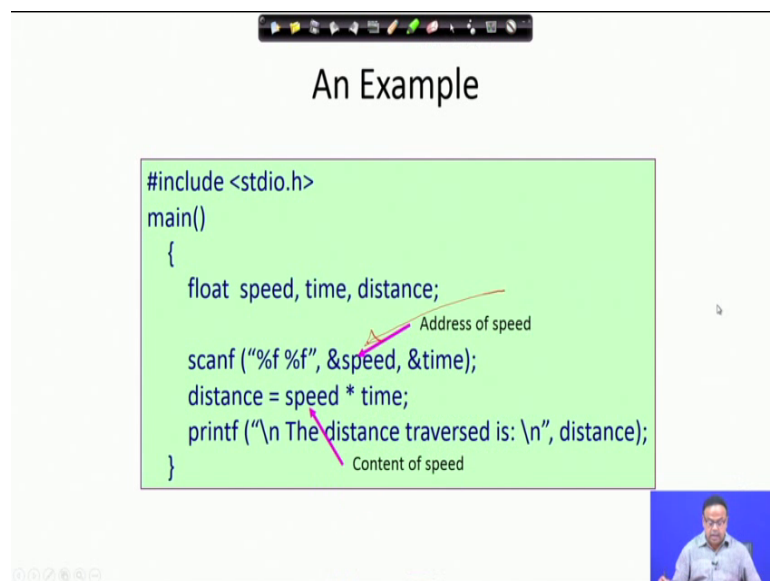


Problem Solving through Programming In C
Prof. Anupam Basu
Department of Computer Science & Engineering
Indian Institute of Technology, Kharagpur

Lecture – 11
Assignment Statement and Operators in C

In the last lecture, we encountered a special notation as if like this where we can put in ampersand and a variable name to denote the address of the particular variable right.

(Refer Slide Time: 00:24)



An Example

```
#include <stdio.h>
main()
{
    float speed, time, distance;

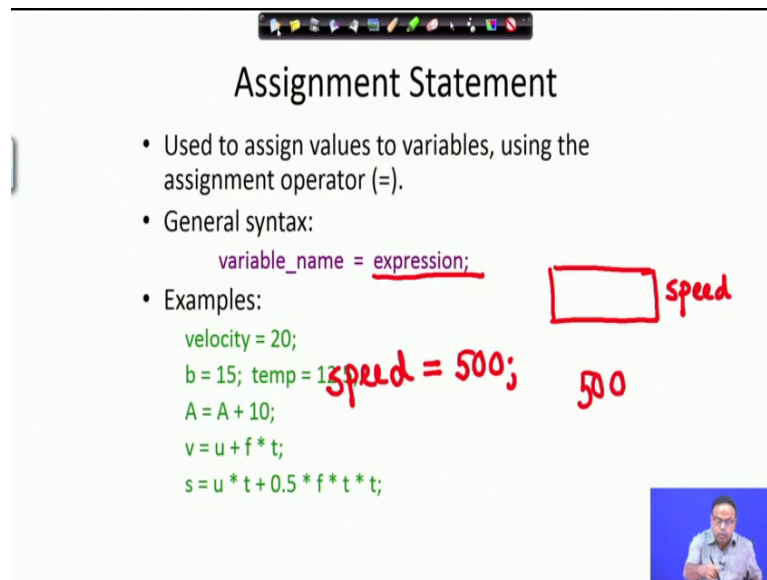
    scanf ("%f %f", &speed, &time);
    distance = speed * time;
    printf ("\n The distance traversed is: \n", distance);
}
```

Address of speed

Content of speed

So, that is what we encountered in the earlier class. Next we will be now we will be moving ahead towards some statements which are absolutely essential for writing any C program.

(Refer Slide Time: 00:54)



Assignment Statement

- Used to assign values to variables, using the assignment operator (=).
- General syntax:
`variable_name = expression;`
- Examples:
`velocity = 20;`
`b = 15; temp = 1;`
`A = A + 10;`
`v = u + f * t;`
`s = u * t + 0.5 * f * t * t;`

`speed = 500;` (with a red box around 'speed' and '500' written next to it)

Just as in English, we write different types of sentences, similarly in the language C just as it is a language there are again different types of statements that we can write, and through which we can express what we want to do. The simplest type of statement that we have already encountered about is assignment statement. Assignment statement means I have got some contents in some resistor; some resistor say X some memory location say variable x and has got some value 50, now how does this value come over here? Initially it was not there. So, suppose I have got memory location variable say speed, and I want to put in some value into this location alright say 500. So, what we do we write speed assigned 500 semicolon.

So, the general syntax for dot is a variable name followed by the expression; as you have written here speed is a variable name and the expression here is nothing, but the constant alright it could be something else.

(Refer Slide Time: 02:50)

Assignment Statement

- Used to assign values to variables, using the assignment operator (=).
- General syntax:
`variable_name = expression;`
- Examples:
`velocity = 20;`
`b = 15; temp = 12.5;`
`A = A + 10;`
`v = u + f * t;`
`s = u * t + 0.5 * f * t * t;`

Handwritten annotations in red:

- Arrows pointing to the variable names in the first three examples.
- A vertical line separating the printed examples from the handwritten ones.
- Handwritten examples: `speed = 2 * 500;` and `speed = v * q;`

Small video inset in the bottom right corner showing a person speaking.

For example, I could have written speed assigned 2 times 500 or may be v times q whatever speed is assigned v times q. I and here are some examples of such assignments. As you can see here velocity is a variable right on the left hand side the variable name is there, on this side is an expression now what type of expression is this? 20 is nothing, but a constant value. Please note that all the statements must be ended with a semicolon you get the second. B assigned 15 again a constant. Look at third one temp assigned 12.5. So, what type of variable is temp? You will immediately answer temp must be a floating point variable of real number.

Here you see a different type of expression a variable A is being assigned the variable A plus 10 what does it mean. So, A assigned A plus 10. So, A being a variable, A is a memory location and suppose it has got some value 25.7, if A is a floating point variable.

V is one variable, u is one variable I am sorry I am using 4 variables here, f is another variable and t is another variable.

Now, suppose you have got some value 20 f have got some value 0.5, and t has got some value 2. Then v is being computed as u is been taken 20 plus the product of these 2 2 and 5. So, 2 and 0.5 will be 1 and 1 is being added to 20. So, these 2 being added is becoming 21, and this 21 is filling up this variable v alright. So, here you see I can use more number of variables, I am sorry this is must be a little nice. So, it should be 21.0. Similarly here you see if they mix up variables how many are here s if a variable u is a variable t is a variable f t t, f t. So, 1 2 3 are there again u f t and s alright and there is a constant 0.5. So, this side entirely this side is the expression.

Similarly, this is again an expression, this is an expression this is also an expression will see more of these expressions in a moment alright.

(Refer Slide Time: 07:56)

Contd.

- A value can be assigned to a variable at the time the variable is declared.

accn

float accn; }
accn = 2.5; }

float accn = 2.5; ✓

So, a value can be assigned to a variable when the variable is declared for example, when I am declaring a variable, just starting I want to declare a particular variable and. So, I want to declare a variable may be acceleration. Acceleration alright now when and I say that acceleration is a real number. So, when I declare it I will write float acceleration, and then at some point later on I can say acceleration assigned say 2.5, this is one way. The other way is that I could have written it when I declared it a float acceleration assigned or initialised to 2.5 this is also allowed in c. So, a value can be assigned when it is declared.

(Refer Slide Time: 09:13)

Contd.


- A value can be assigned to a variable at the time the variable is declared.

```
int speed = 30;  
char flag = 'y';
```

*char flag;
flag = 'y';*
- Several variables can be assigned the same value using multiple assignment operators.

```
a = b = c = 5;  
flag1 = flag2 = 'y';  
speed = flow = 0.0;
```

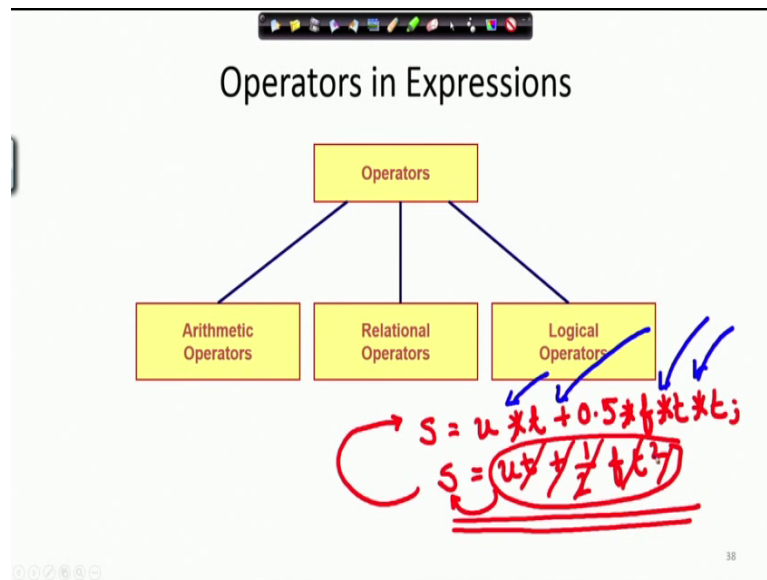
float speed = flow = 0.0;



For example here speed is an integer, which is being declared as an integer and along with that it is being assigned the value 30. Here you see the char is a type of another variable flag and when I am saying that it is a flag is a character, I am along with that I am also assigning it to value y.

Now, you understand you remember that within the single quote means it is a did the character string. So, I could have done it also like this char flag, and then alter on flag assigned y that is equivalent to what i did here. Several values can be assigned the same value using multiple assignments operative what does it mean? As soon as we see the example it will be clear for example, a b c all these are being assigned the value 5 flag ne and flag 2 both are being assigned the value y, speed and flow both are being assigned the value 0.0. So, here say for example, the correct thing would be to say float, speed flow are assigned 0.0. So, that is the simplest opposable way we go about it.

(Refer Slide Time: 11:05)



Now, when we write an expression, first thing we have seen now is the assignment statement. Now in the assignment statement if you have seen the earlier slide we were using some operative. For example, u assigned say v minus f times t right. So, these are an; this is an assignment statement here is an assignment, but on this side I have written an expression.

So, how can we write an expression in c. We have seen quite a few examples of that expressions in the earlier slides right, we have seen different types of expressions like s assigned u times t plus 0.5 times f times t times t. So, this is you will immediately recall that this is our standard school formula f is equal to u t plus half f t square right. Now this expression has got 2 components, this well-known expression in school has got 2 components. One side is the expression here which has to be computed and then that has to be assigned to another result of other variable. So, that is being written here. Now this is the way an expression is written in c you cannot just write it in this way as we are doing in our school, this is not possible we have to write it in this way ok.

Now, here when we write this expression you please observe a couple of things, look at these this, this, this. These are known as the operators you have this is meaning multiplication this is meaning addition these are again multiplication symbols. Now these they are sets of allowed operators in c. Just as every language allow some contrast to form sentence and some of them are not some other contrasts are not valid in a

sentence formation, similar to that in the programming language c or for that matter for any programming language there are some allowed operators by which we can form expressions. So, we have got 3 types of operators the type of operators; that we have encountered till now we have seen till now are arithmetic operators, but besides arithmetic operators there are 2 other types of operators called relational operators and logical operators ok.

(Refer Slide Time: 14:30)

Arithmetic Operators

- Addition :: +
- Subtraction :: -
- Division :: /
- Multiplication :: *
- Modulus :: %

Handwritten annotations on the slide include blue arrows pointing to the operators, a handwritten division symbol (÷) and multiplication symbol (×), and a handwritten modulus calculation: $15 \% 3 = 0$, with a long division diagram showing $3 \overline{)15} \ 5$ with a remainder of 0.

So, let us see a little more of this arithmetic operators, these are the some of the very familiar arithmetic operators you know that this is, this means addition normal subtraction, this is division, remember unlike this sort of division we use in schools we use here this symbol for division. Unlike this symbol that we use for multiplication, here we use this symbol and new symbol that we are introducing here is this symbol, like the percentage sign. Now this does not mean percent computing percentage, it means modulus what does modulus mean? Say the modulus means finding the remainder for example, if I compute 15 modulus 3; that means, I am dividing 15 by 3 and whatever is the remainder is my result. So, what is my remainder here? Remainder is 0. So, modulus is basically the remainder operator. So, another example let us look at.

(Refer Slide Time: 15:56)

Arithmetic Operators

- Addition :: +
- Subtraction :: -
- Division :: /
- Multiplication :: *
- Modulus :: %

Handwritten notes:

- quotient = 13
- remainder = 1
- $27 \% 2 = 1$
- $26 \% 2 = 0$
- $26 / 2 = 13$

The slide includes a small video inset of a presenter in the bottom right corner.

Say I have got 27 modulus 2 what would be the result be? 27 divided by 2 the quotient is 13 right quotient is 13 and the remainder is 1. So, this modulus would be 1 whereas, if I has done 20 6 modulus 2 that would be 0.

Now, again I have got if I instead of modulus operator if I had done 26 divided by 2 the result would be 13. So, this gives you the quotient whereas, this gives you the remainder alright. So, this is a new operator that we are coming across and you should keep that in mind next let us proceed.

(Refer Slide Time: 17:05)

Examples

```
distance = rate * time ;
netIncome = income * tax ;
speed = distance / time ;
area = PI * radius * radius ;
y = a * x * x + b * x + c ;
quotient = dividend / divisor ;
remain = dividend % divisor ;
```

Handwritten notes:

- #define PI 3.1415...
- Area = πr^2

The slide includes a small video inset of a presenter in the bottom right corner.

And here some examples distance is rate or velocity or speed multiplied by time. Please note again as I have told earlier also that any expression must end with a semicolon as being done here what does it this mean? Can you read this variable, can you read this variable it is meaningful net income is income minus tax. So, operator is minus this is the arithmetic expression and this is the arithmetic operator. Speed is distance divided by time again ended with a semicolon speed is distance divided by time how do we find the area of a circle? Pi you remember pi we can define pi we had seen this example earlier has defined pi 3.1415 etcetera, etcetera, we could have done that. So, pi is a constant times radius times radius. Here basically what I am computing is area is being assigned is the assignment operator and the expression is pi r square. Now this expression I am writing in this way pi times r radius times radius.

(Refer Slide Time: 18:50)

Examples

```

distance = rate * time ;
netIncome = income - tax ;
speed = distance / time ;
area = PI * radius * radius;
y = a * x * x + b * x + c;
quotient = dividend / divisor;
remain = dividend % divisor;

```

$y = ax^2 + bx + c;$

Here is another expression y assigned a x square what does it, how does it how do we write it typically in school. This is expression which is a x square plus b x plus c alright this is a very familiar expression of a quadratic expression. Now when we write it again here you see how many operators I have got 2 multiplication operations 3 multiplication operations and 2 addition operations right and one assignment operation. So, here again quotient is divided by divisor. Now this is exactly what I was telling a couple of moments back, that this operator is actually turning you the quotient alright of a division operation and this is actually giving you the remainder of a division operation. So, here are some examples of arithmetic expression.

(Refer Slide Time: 20:06)

Contd.

- Suppose x and y are two integer variables, whose values are 13 and 5 respectively.

$x + y$	18
$x - y$	8
$x * y$	65
x / y	2
$x \% y$	3

Handwritten annotations:

- A box containing the value 65.
- A box containing ~~13~~65, with an arrow pointing to the x variable.
- A box containing 5, with an arrow pointing to the y variable.
- Equation: $z = x * y$
- Equation: $x = x * y$

41

Suppose x and y are 2 integer variables, and the values are we know 13 and 15. When added to y x plus y arithmetic operator plus will give me 18 x minus y 13 minus 5 will give me 8. Now the point to note here is that here again always try to think in terms of our memory location diagram x and y are 2 variables, x is 13 and y is 5. So, x plus y means the content of the location x plus the content of the location y ; x minus y is the content of the location x minus the content of the location y .

Similarly, when we multiply it is 65, now if I add an assignment with this operation with this operation I just add a assignment for example, I write I am giving 2 variations z assigned x times y ; that means, what? There is another location with where the content of x and the content of y are taken and multiplied and 13 times 5 is 65 that is stored there and that is possible because I have assigned it here. Again I could have assigned alternative could have done x assigned x times y what would have happened in this case? The content of x would have been taken 13 multiplied with a content of y that is 5 13 and 5 would be 65, and this product 65 where would that be written? It would be written in x why because it is being assigned to x . So, then this would be over written with 65.

Similarly, we will recall now x divided by 2 is a division from expecting the quotient 13 divided by 5 what will be the quotient 2, but x modulus y would be the remainder of when I divided 13 by 5. So, that modulus is 3.

(Refer Slide Time: 22:50)

Operator Precedence

- In decreasing order of priority
- 1. Parentheses :: ()

$x = (p+q)*z - x/(l+m);$

$A * z - x / B$

The slide includes a toolbar at the top and a small video inset of a presenter in the bottom right corner.

Now, in an expression we can have different operators. If more than one different operator occurs in a particular expression how will that expression be evaluated. So, now, we are concerned about how we will evaluate or find the result of computing an expression k. Now here is a list in decreasing order of priority. So, if I have something like this say x assigned p plus q times z minus x divided by l plus m. Now here you can see that I have got different operators what are those? Our well known operators are plus multiplication minus division plus again here is another operator that is parentheses. Now as we learnt in school algebra that the parentheses has got the highest priority alright. So, I will first compute the elements which are within the parentheses. So, first p plus q will be computed then l plus and l plus m will be computed now out of this p plus q and l plus m which one will be computed first? Whenever these 2 that are these are of the same priority this parentheses, now if there will be more than one operator of the same priority they will computed left to right.

So, first we will have p plus q computed alright suppose that is something, say let us call it a some value A times Z minus X suppose this is computed to be some constant B. So, in that way it will be computed. So, the parentheses has got the highest priority next after parentheses is unary minus.

(Refer Slide Time: 25:20)

Operator Precedence

- In decreasing order of priority
 - Parentheses :: ()
 - Unary minus :: -5

$x - y$ $x = 2.5;$

$-x + 7.5;$
 $-2.5 + 7.5;$
 5.0

$-2.0 + 6.0$

Unary minus means usually when we write something like x minus y , then I have got 2 variables on which I am carrying out this subtraction. This is binary operator in the sense that I am leaving 2 variables of 2 constants 2 elements and which I am carrying out this computation. Unary minus means that particular variable is being operated on for example, if I had something like this minus x plus 7.5 semicolon and suppose x was 2.5. So, what will be the value? x is 2.5. So, for this will be done. So, it will be minus 2.5 plus 7.5. So, the result will be 5.0; it is not that I will first compute this 2.5 plus 7.5 and then do negation alright.

So, the unary minus. So, here I have shown a variable, it could be something constant also minus 2 plus 6 that means, you all know that 6 subtracted by 2 although it is plus because this has got the higher precedence.

(Refer Slide Time: 27:17)

Operator Precedence

- In decreasing order of priority
 1. Parentheses :: ()
 2. Unary minus :: -5
 3. Multiplication, Division, and Modulus

$x/y * z/q$

42

The next one is multiplication division and modulus. These 3 will have the same priority multiplication division and modulus these 3 operators will have same priority therefore, if I have an expression like say x division y multiplied by z modulus q how will that be which one will be done first? This, this and this have got the same priority therefore, we will carrying it out left to right alright, but if the expression was something like this minus.

(Refer Slide Time: 28:06)

Operator Precedence

- In decreasing order of priority
 1. Parentheses :: ()
 2. Unary minus :: -5
 3. Multiplication, Division, and Modulus

$-x/y * z/q$

42

X divided by y multiplied by z modulus q then which would be done first? First is unary minus should be done first followed by these 3 candidates which will be done left to right left to right.

(Refer Slide Time: 28:42)

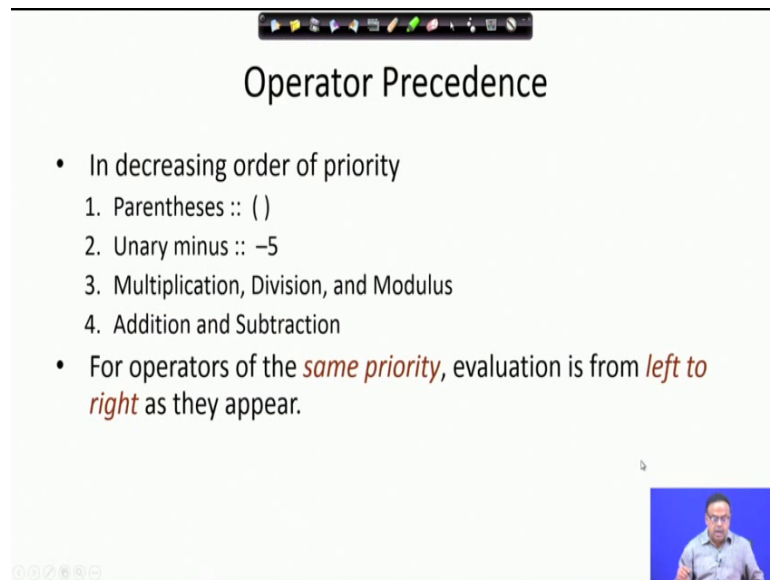
Operator Precedence

- In decreasing order of priority
 1. Parentheses :: ()
 2. Unary minus :: -5
 3. Multiplication, Division, and Modulus
 4. Addition and Subtraction

$p + x * y / z - q$
 \downarrow
 $p + A / z - q$
 \downarrow
 $p + B - q$
 \downarrow
 $c - q$

Next comes addition and subtraction. So, if I have got an expression again like say x times y even before that if we put it, p plus x times y divided by z minus q then in which order would it be computed? First plus will not be computed multiplication has got a higher priority this also has got an higher priority. So, out of these 2 which one will be done first left to right? So, first I will do this suppose this is yielding a result A. So, it will be p plus A then divided by Z minus q and then this one will be done because left to right of the same precedence suppose it is B then it turns out to be p, it turns out to be p plus b minus q alright. Now out of this plus and minus have got the same priority then which one will be done first this part will be done first, suppose that is c, c minus q this is how this entire operation will be done.

(Refer Slide Time: 30:12)



Operator Precedence

- In decreasing order of priority
 1. Parentheses :: ()
 2. Unary minus :: -5
 3. Multiplication, Division, and Modulus
 4. Addition and Subtraction
- For operators of the *same priority*, evaluation is from *left to right* as they appear.

For operators of the same priority the evaluation is from left to right. In the next class we will see some more examples of this and we will proceed further.