

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

Lecture – 29
Program Using Arrays

So, we saw a program where we are finding the minimum of a set of 10 numbers.

(Refer Slide Time: 00:22)

Example 1: Find the minimum of a set of 10 numbers

```
#include <stdio.h>
main()
{
    int a[10], i, min;
    printf("Give 10 values \n");
    for (i=0; i<10; i++)
        scanf ("%d", &a[i]);

    min = 99999;
    for (i=0; i<10; i++)
    {
        if (a[i] < min)
            min = a[i];
    }
    printf ("\n Minimum is %d", min);
}
```

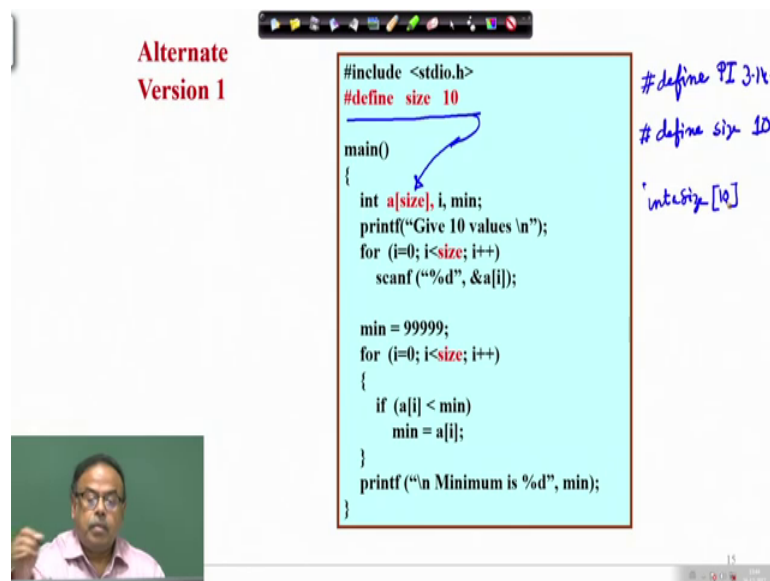
So, here the program is restricted as you can see to 10 numbers and accordingly I have declared the array a to be of size 10 and also this iteration values are also iteration limit has been kept to 10. Now, when we did that just a quick recapitulation of what we did in the last lecture that here is the array declaration and this is where we are accessing the array elements. Here you can see I am accessing the array element a i, a particular array element here and assigning that to another variable mean.

Now, here it is very important how am I reading the element. Now, since an array consists of a number of elements sorry, the since the array consists of a number of elements I have to read them in a loop that is a very fundamental thing that you one should understand that at a time I cannot read the entire array in one shot, except for the case where I initialize it and when I declared the array I give int a, so and so values within curly bracket that is one way. But if you have to do it at runtime; that means, whenever the you asking the program is running and you asking from the user that you

entered the values then you have to take one bit of value at a time just like this and store it in the array.

So, that has to be done in a loop for by varying this indexes one after another as the index moves I load the different locations with the values. Now, given this we can have an alternate version of the same program.

(Refer Slide Time: 02:26)



Alternate Version 1

```
#include <stdio.h>
#define size 10

main()
{
    int a[size], i, min;
    printf("Give 10 values \n");
    for (i=0; i<size; i++)
        scanf("%d", &a[i]);

    min = 99999;
    for (i=0; i<size; i++)
    {
        if (a[i] < min)
            min = a[i];
    }
    printf("\n Minimum is %d", min);
}
```

#define PI 3.14...
#define size 10
int a[size]

You can see that here we have added one line that is this line, define size 10. This is again the use of hash define that we had learnt when we defined pi to be 3 point we had encountered this defined pi 3.14 etcetera etcetera. So, similarly here I am defining size a variable size to have the value 10, therefore, I now, can write this dimension as size. Now, please note that the compiler look at it from the angle of the compiler, the compiler will try to allocate some memory location some amount of memory location ok. Now, if there be a variable size it really does not know how much memory location to allocate, but since I have defined it earlier it now, knows size means 10. So, it will replace int size with int size sorry, int a size to be int a size 10.

Now, next thing you do is wherever I had i less than 10 I make it size i less than size here also for the for loop I make i is less than size. Why did I do that? What is the advantage of doing it? Now, suppose you have got this program and how many places is size being used it is being used in 1 2 3 places in another program it could have been used in more

number of places. Here we have not printed the array, I have only printed the minimum. So, there could be more places where sites could be used.

Now, suppose I want to modify this program, or reuse this program for an array of size 100 only thing I need to do is I change this 10 to 100. And please note that after defined any hash define does not have a semicolon because these are not part of the program, these are commands to the compiler or pre processing commands.

(Refer Slide Time: 04:55)

Alternate Version 1

```
#include <stdio.h>
#define size 10
main()
{
    int a[size], i, min;
    printf("Give 10 values \n");
    for (i=0; i<size; i++)
        scanf ("%d", &a[i]);

    min = 99999;
    for (i=0; i<size; i++)
    {
        if (a[i] < min)
            min = a[i];
    }
    printf ("\n Minimum is %d", min);
}
```

Handwritten annotations on the slide:

- A blue arrow points from the handwritten "100" to the "10" in the `#define size 10` line.
- A blue arrow points from the handwritten "a[100]" to the `a[size]` in the `int a[size]` line.
- Handwritten "i < 100" is written next to the `i < size` in both the first and second `for` loops.

So, I if I could make it 100 then this would be replaced by 100 it would be a 100 and here it should be i less than 100 automatically, i less than 100. I need not change it at all the places that is the advantage of the hash define.

(Refer Slide Time: 05:42)

Alternate Version 1

Change only one line to change the problem size

```
#include <stdio.h>
#define size 10

main()
{
    int a[size], i, min;
    printf("Give 10 values \n");
    for (i=0; i<size; i++)
        scanf("%d", &a[i]);

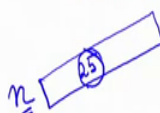
    min = 99999;
    for (i=0; i<size; i++)
    {
        if (a[i] < min)
            min = a[i];
    }
    printf("\n Minimum is %d", min);
}
```

15

So, we change we can change only one line to change the problem size. The problem size from 10 to 100 to 1000 we can change by just changing one line.

(Refer Slide Time: 05:56)

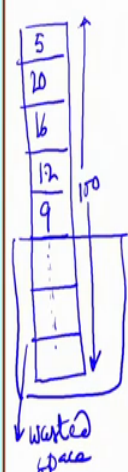
Alternate Version 2



```
#include <stdio.h>

main()
{
    int a[100], i, min, n;
    printf("Give number of elements (n) \n");
    scanf("%d", &n); /* Number of elements */
    printf("Input all n integers \n");
    for (i=0; i<n; i++)
        scanf("%d", &a[i]);

    min = 99999;
    for (i=0; i<n; i++)
    {
        if (a[i] < min)
            min = a[i];
    }
    printf("\n Minimum is %d", min);
}
```



16

Here is another version of the program. Here we have not defined hash defined, but I have kept a large size 100. Now, so, I have got an array space of 100 elements all right 100 elements. So, much space is kept for me by this line. Now, what I am doing is I am asking from the user you please tell me how many elements you are going to input. Suppose here and that I am taking as a variable n, n is a variable where I am just asking

the user give the number of elements n and when the user is typing say 25 then that goes as n. So, out of now, this 25 is of course, less than the size of this which is 100 it is less than that.

So, now, I will just work not for 100 elements, but for 25 elements why because the user has already told me that I want to input 25 elements. Therefore, what modification should I do? Again here I want to make it interactive. So, I again tell the user on the screen you will see this message input all in integers and the user is entering them one after another. He can type in 25 enter 35 enter 42 enter like that or he can give space and go on typing that.

So, for i less than 0 to n, now what is n? n is 25. So, now, it is for i to i, 0 to 25 I read the number. So, I will be reading 25 numbers here 5, 20, 16, 12, 9 etcetera etcetera I go on and I will go up to 25 not 100 all these. So, this space will be wasted, there is a wasted space. Why wasted space? Because I had deserved 100 locations, but the user has just agreed to give only 25 numbers and then the remaining part remains the same here also I will be dealing with n. Why n? I will be searching. So, when I did this the array that I got is actually although they were 100 elements the array was actually of 25 elements right, the array was of 25 elements. So, I will have to find the minimum within this zone and I need not go here.

(Refer Slide Time: 08:57)

Alternate Version 2

Define an array of large size and use only the required number of elements


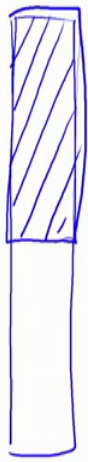
```
#include <stdio.h>

main()
{
    int a[100], i, min, n;

    printf("Give number of elements (n) \n");
    scanf ("%d", &n); /* Number of elements */

    printf("Input all n integers \n");
    for (i=0; i<n; i++)
        scanf ("%d", &a[i]);

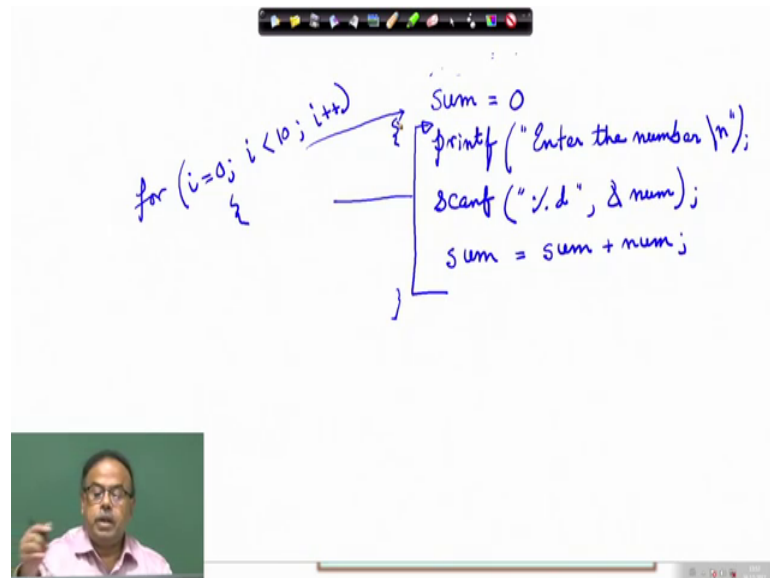
    min = 99999;
    for (i=0; i<n; i++)
    {
        if (a[i] < min)
            min = a[i];
    }
    printf ("\n Minimum is %d", min);
}
```



16

So, that is another version of the same program. So, here what is done is we define an array of larger size and only use the required amount out of that all right. Now, let us do a programming ourselves. Suppose I want to read the some integers and I want to find out the sum of all the integers that old example that we had done earlier. How did we do it earlier let us recall.

(Refer Slide Time: 10:03)



We did let me write it in this way sum there was something like sum assign 0, then printf, enter the number, then scanf percentage d and num sum equals sum plus num and then this has to be done in a loop ok.

So, watch how can I make a loop out of this. So, I also read another variable the number of numbers. So, sum n was read here. So, I had done some scanf. I would say let us say of 10 numbers. So, I need not scanf. So, this thing I have to loop right this part I have to loop how long. So, what should I put here? At the beginning for doing the loop for i, since I know beforehand that there are 10 numbers, i assign 0, i less than 10, i plus plus then I put a parenthesis here and a parenthesis here. So, and the first statement comes here. Confusing? Should I write it again or is it clear.

(Refer Slide Time: 12:07)

```
sum = 0;
for (i = 0; i < 10; i++)
{
    printf("Please enter the number\n");
    scanf("%d", &num);
    sum = sum + num;
}
```

So, I can write for i assign 0, i less than 10, i plus plus, printf please enter the number backslash n, then here I had done sum assign 0 scanf percentage d and num sum assigned sum plus num that is how the loop will go on. But here I am not remembering the number.

So, what I can do using an array what I can do is, let me now do it using an array int, num say 10, comma i is another integer variable. Now, and also I need sum sorry sum is also another integer, sum.

(Refer Slide Time: 13:26)

The slide shows handwritten code and a diagram. The code is as follows:

```
int num [10], i, sum;
/* Read the numbers */
for (i = 0; i < 10; i++)
{
    printf("Enter number\n");
    scanf("%d", &num[i]);
}
/* Add the elements */
sum = 0;
for (i = 0; i < 10; i++)
    sum = sum + num[i];
/* Find average */
avg = sum / 10;
```

The diagram illustrates an array of 10 elements. The first four elements are 5, 4, 3, and 2. The sum of these four elements is calculated as 5 + 4 + 3 + 2 = 14. The total sum of all 10 elements is given as 25. The average is calculated as 25 / 10 = 2.5. The word 'avg' is circled in the code.

Now, I will first read the array, the first is reading the array. So, let me put a comment read the numbers all the numbers earlier you see the scanf every time I was doing scanf I was reading the number. Here I am reading the numbers and storing them.

So, what should I do? For $i = 0$ to $i < 10$, $i++$ let us increment the array. What am I doing? `printf` enter number `\n` `scanf`. Now, what should I do? I am reading an integer, so percentage `d` and ampersand. What should it be? What am I reading first? Here is an array whose name is `num`. So, what is the first thing that I will be reading? First thing that I will be reading is `num[0]` this element, so `scanf` percentage `d` and `num[i]`. Why `i`? Because `i` is the index and I can see that initially `i` is 0. So, whatever is being scanned will be stored here then I will go back in this loop then the next one will be stored then the next one will be stored in this way the entire array will be stored after that I am trying to find the sum. So, now add the next part that I do is add the elements.

Now, in this here I will have to again add the elements. So, suppose the, suppose the array that has been formed has got some elements like say 5 7 13 12 6 so and so forth. So, this I have to add this with this then I will get a sum and with that 12, I will be adding 13. So, I will be getting 25, then 25 and 12, 37 in that way I will go on.

So, here I initialize `sum` to be 0 and then for again I assign the same index I can use, I could have used other index also $i < 10$ sorry $i++$. I hope you remember that after each of these we put a semicolon right in for loop, some assigned some plus `num[i]`. So, what will happen? `sum` was 0, `sum` is 0. First, in the first iteration of the loop what will happen? I take my value of `i` is 0. So, `num[i]` will be taken and what is `num[i]`, the first element of the `num` array. So, that is 5 `sum` plus 5. So, `sum` will be 5.

Now, next I go back here `i` becomes 1. So, `i` come to the second element and `i` add that with `sum`. So, `sum` becomes 12 automatically go there I take the index is updated, `i` is being updated here. So, I take the third element and add with `sum`, so it becomes 25 and in this way I go on adding them all the elements in an array. So, the same thing I could do using, but in this case what is the advantage that I am getting? I have got, I have not forgotten the numbers that I have read. So, I can do that and again I can print all these numbers I can say now, the sum of these numbers is so much.

So, that is how we can use an array. Now, suppose I had, I had to find the average of the elements what should I do? I have got the `sum` here. And then find average, what should

I do here? Here of course, I will need another variable average and so I can say and average is sum divided by 10. There is a mistake that is there. What is the mistake can you find out? Here I found sum that is ok, and then I am finding the average and why I am why while I am finding the average I am dividing some whatever sum I got here by the number of elements which is 10, I knew beforehand that these are 10 elements. Where is the error? What is the mistake? The mistake is that the average cannot be need not be an integer right. Suppose my sum was 25 and I divide by 10, so it to be 2.5. Therefore this average should not be kept here I should have declared it as a float average and not with the int right. So, you must be very careful about it. So, this is how what we have shown here is how we can read an array and how you can use an array.

(Refer Slide Time: 21:25)

Example 2:
Computing gpa

Subject	grade point	credit
0	3	5
1	5	5
2	4	4
3	5	3
4	3	4
5	5	4

```

#include <stdio.h>
#define nsub 6 /* number of subject */

main()
{
    int grade_pt[nsub], cred[nsub], i,
        gp_sum=0, cred_sum=0, gpa;

    printf("Input gr. points and credits for six subjects\n");
    for (i=0; i<nsub; i++)
        scanf("%d %d", &grade_pt[i], &cred[i]);

    for (i=0; i<nsub; i++)
    {
        gp_sum += grade_pt[i] * cred[i];
        cred_sum += cred[i];
    }

    gpa = gp_sum / cred_sum;
    printf("\n Grade point average: is %d", gpa);
}

```

Handwritten calculations on the right side of the slide:

$$\begin{array}{r}
 15 \\
 +25 \\
 +16 \\
 +15 \\
 +12 \\
 +20 \\
 \hline
 103 \\
 25 \\
 \hline
 = 4.12
 \end{array}$$

So, let us look at this example of computing the grade point average of the students. Let us try to understand what this program does.

We have defined a variable n sub to be 6, this n sub means number of subjects all right. Now, there are 6 subjects. So, 0, 1, subject 0, subject 1, subject 2, subject 3, subject 4, subject 5, now for each of them we are taking a great point which is nothing, but a mark say, an integer 3 or 5 or 6 whatever, for every how many 6. Here you see I have used define so this line essentially means great point, 6, for 6 subjects and credit for each subject. So, suppose I have got the subject and the great point and each subject has got a credit and I have got 6 subjects.

So, subject 0 has got a great point, he has got a great point of 3, maybe he has got a b grade or whatever and the credit; that means, the importance of that subject is 5, subject 1 he has got a great point of 5, he did really well here and the credit of that was also 5. There was another subject 2 in which his great point was 4 and the credit of that subject was 4 that is the importance weightage. The subject 3 he got 5 and the credit was 3 like that and since we have got 6 subjects let us do that. Subject 4 his credit he got 3 and the importance was 4 and this one he got 5 out of 5 and the credit was again 4.

Now, the credit point is the great point is computed here as for each of them we will multiply the great point with the credit. So, it will be 3 times 5; that means, 15 plus 5 times 5 that is 25 plus 4 times 4 that is 16, then 15, then 12, then 20 and all these are added up. So, 40, 56, 66, 71 and 12, 71 and 12 is a 83 and 20. So, he got 103 points. And how many credits were there? 5, 5, 10, 10 and 7, 17 and 8, 25.

So, his grade point average gpa will be when I divide this by 25 his thing will be 4 point something, 4.1 etcetera. So, 4.15 or something like that. So, this is what I want to compute.

(Refer Slide Time: 25:42)

Example 2:
Computing gpa

$$\frac{\sum \text{credit}_i * \text{gpi}}{\sum \text{credit}_i}$$

```

#include <stdio.h>
#define nsub 6

main()
{
    int grade_pt[nsub], cred[nsub], i,
        gp_sum=0, cred_sum=0, gpa;

    printf("Input gr. points and credits for six subjects \n");
    for (i=0; i<nsub; i++)
        scanf ("%d %d", &grade_pt[i], &cred[i]);

    for (i=0; i<nsub; i++)
    {
        gp_sum += grade_pt[i] * cred[i];
        cred_sum += cred[i];
    }

    gpa = gp_sum / cred_sum;
    printf("\n Grade point average: is %d", gpa);
}

```

Therefore, what I need is to find out that, I have to find the sigma of the sum of the credit for a course i times the great point of that course i, I will do it for all the courses and then the whole thing should be divided by the total number of credits, total number of credits.

Now, let us see how we have done, we can do this program. Here we can find that the great point for 6 and credit for 6 and there is I the great points sum. So, I have to do this sum is 0 credit sum is 0 and I have to find out a gpa. So, what I am asking from the user here input great points and credits for the 6 subjects.

(Refer Slide Time: 26:43)

The image shows a C program titled "Example 2: Computing gpa" with handwritten annotations. On the left, there are two arrays: "gp" with values [3, 5, , , ,] and "cr" with values [4, 3, , , ,]. Below them is the formula: $gp_sum = gp_sum + gp[i] * credit[i]$. On the right, the code is shown with annotations: "Reading 2 arrays" points to the scanf statement, and $\sum gp[i] * cred[i]$ and $\sum c[i]$ point to the accumulation lines in the loop.

```

#include <stdio.h>
#define nsub 6

main()
{
    int grade_pt[nsub], cred[nsub], i,
        gp_sum=0, cred_sum=0, gpa;

    printf("Input gr. points and credits for six subjects \n");
    for (i=0; i<nsub; i++)
        scanf("%d %d", &grade_pt[i], &cred[i]);

    for (i=0; i<nsub; i++)
    {
        gp_sum += grade_pt[i] * cred[i];
        cred_sum += cred[i];
    }

    gpa = gp_sum / cred_sum;
    printf("\n Grade point average: is %d", gpa);
}

```

So, where are those being stored? Those are being stored in two different arrays. Here is the great point array, this thing and there is another credit array 6 subjects.

So, I read for each of the subjects i less than, i 0 to n sub, I read the great point say 3 and the credit for that subject say 4. And again I go and store the great point of that subject great point that is been obtained is 5 and say credit is 3. In that way I go on filling up reading 2 arrays. So, here I am reading two arrays you see, here I am reading two arrays in one scanf statement and through one for loop, through one for loop I am filling up these two arrays. So, the reading part is complete here.

Next I come to the computation. Grade point sum will be you, now know what is meant by this. This means grade point sum is equal to grade point sum plus grade point i times credit i . So, it is gp sum is whatever is the gp sum initially gp sum was made 0, gp sum plus grade point i times the credit of i and that has to be done in a loop just as we had added the elements of an array. So, and credit sum also I am also finding out the credit sum the credit sum was 0. So, whatever the credits are I am also adding them. So, by adding this array I am getting the total number of credits here and here I am getting the

grade points sum. Now, the grade point I have sum, grade point average is grade point sum by credits sum and then I print out the gpa.

So, please try to understand this part. What I am trying to do is $\sum \text{grade point } i$, times credit i divided by $\sum \text{credit } i$. So, the $\sum \text{credit } i$ is being computed here this is computing $\sum \text{credit } i$. Why? Because this is being done in a loop and here I am doing this part $\sum \text{grade point } i$ times credit i and adding that with grade point sum. So, these are very useful program which illustrates a number of things and in the assignments you will be given more number of programs to do which will give you a very good practice.