Programming and Data Structures
Prof. N. S. Narayanaswamy
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture – 02
Structures, Pointers and Functions

Welcome to the second lecture of the first week of this programming in data structures course offered by NPTEL, and the response to the first lecture has been very nice, I found that people on the forum are very active and there are many questions and exchanges of thoughts. I have a couple of comments to all of you who are where actively participating on the forum, there are small I would not say bugs. For example, loose programming practices.

For example, many of you will recognize the fact that I had kept a range of a certain variable or counter variable to be one more than exactly what is this necessary. Now, these are specifically given, so that when you count the number of comparisons, you will for example, note that those are not necessary. So, therefore you should trust in your understanding and use your logic to convince yourself that your approaches indeed correct, that is one.

And the second point that I would like to draw your attention to is that read the question carefully and I think the questions are fairly well phrased. And once you understand the question and understand it properly, the answer is fairly straight forward. So, today by the end of this lecture by the time you get to see this lecture, there will be a short description of the first activity which will be available to you on the portal itself on the forums, so I look forward to your comments.

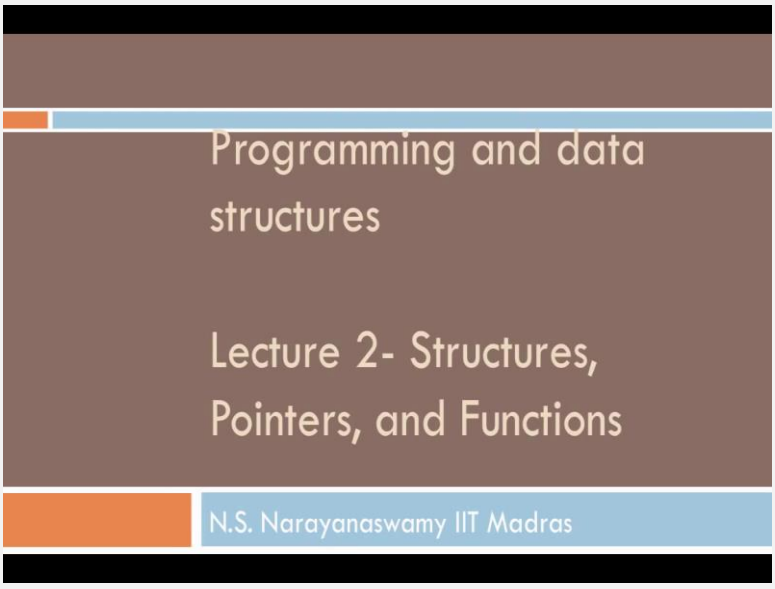
So, many of these are self assessment exercises, they will not carry marks, they do not carry weightage for the final evaluation. So, you do not have to worry about how many marks you will get and so on and so forth. And like I keep saying, enjoy the subject and leave the evaluation to significantly late or when you comfortable with the subject material. Having said all these things, let us move on to the lecture for today.

So, we are going to talk about structures, pointers and functions, these form the central pieces in the understanding of how to design and maintain data structures. I repeat, you need to be able to work with structures, you need to be able to work with pointers and you should be able to write functions which will manipulate these structures and pointers, and this is what forms the central, and these reform the central concepts to work with data structures and that is what you will be expose to in today's lecture.

Of course, you may worry about why we are learning all this in C, I think C gives you a flexibility of the richness and the level at which you can program. So, I mean if you work with a feature rich programming language, then you may really not appreciate the challenges of designing your own data structures. On the other hand, C is a very flexible programming language, where you can implement your own data structures at almost any level that you wish to challenge yourself.

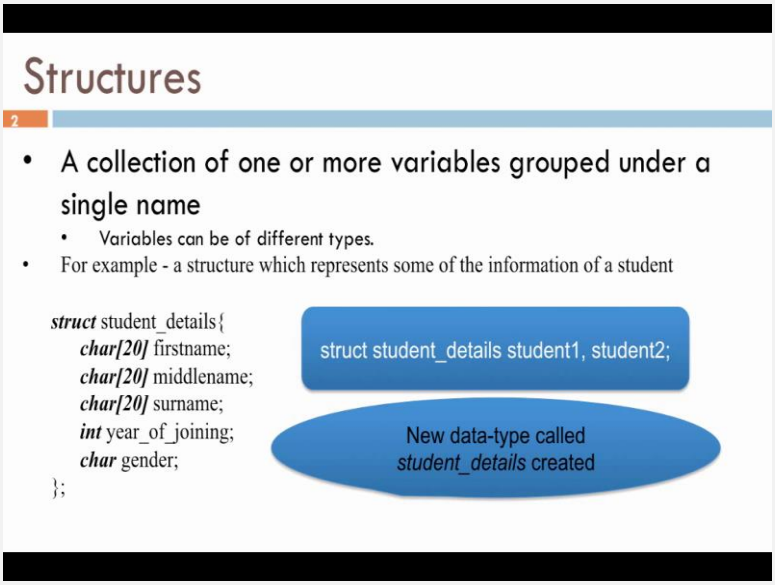
So, we will today's review concepts in C about structures, pointers, and how to manipulate them using functions. Like in the earlier lecture, there will be a part where we will work with real code and hopefully that is useful. So, I look forward to comments on the forum. So, now I am going to take you to the power point presentation.

(Refer Slide Time: 04:32)



So, structures, pointers and functions here we go.

(Refer Slide Time: 04:41)



So, what are structures? Those of you who have already programmed in C to a certain extent will realize that a structure is a syntactic feature given in the C programming language that is, it is a key word, struct is a key word that is given to you on the C programming language, where you can group variables under a single name and these variables can be of different types. As you see on your screen at the moment, you see an example of a structure called student details which has 5 fields, the 5 fields are important has design by the programmer to represents some details about students in general.

And the fields are first name, middle name and surname, which are designed to be character arrays of size 20, there is a bit of a syntax error that has been deliberately introduced there. So, for example here, so where the 20 is associated with the characters with some kind of a pseudo code, just to draw your attention.

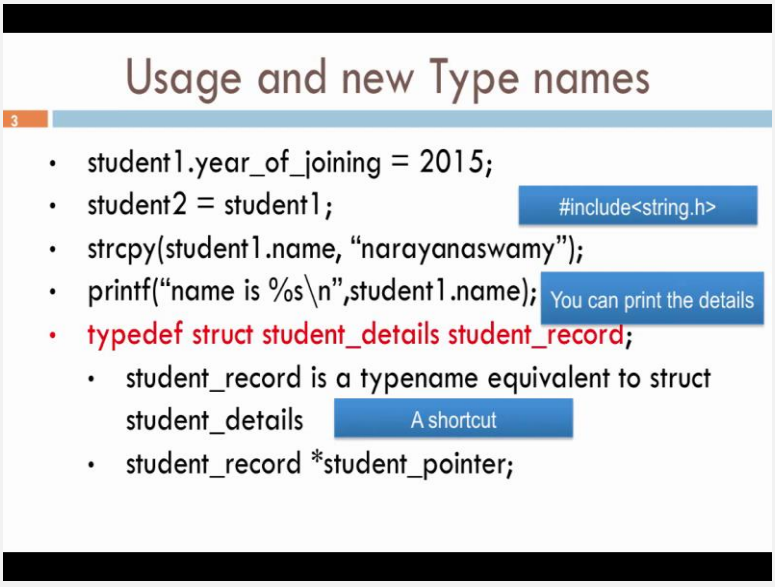
When we look at the program all this is cleaned up, then we maintain an integer which is the year of joining and we also maintain a single character which represents the gender of the student. Now, the most important thing is that as a programmer you think of this as a template for every student. But, as a person who is going deep into programming, the

most important thing is what you see here, a new data type called student details is created by the struct key word in the C programming language. So, therefore using struct you can create a new data type.

Now, after creating a new data type you can define variables of this data type as you can see here; student 1, and student 2 are two variables whose types are struct student details. Therefore, the concept that you need to keep in mind is that we have created a new data type. We will dell deeper into what it is to create a new data type, as we go on in this lecture in these sub sequent lectures. So; however, when I say you have created a new data type, you created a user defined data type putting together variables of different data types.

So, you will naturally have this questions saying, can I put together other user defined data types to create another user defined data type, the answer is yes, I do not have examples for this today, but these will definitely follow in the next few lectures.

(Refer Slide Time: 07:47)



How does one use a new data type? Let us just take a look at this. Let us looked at the simplest of assignments statements, where to the field year of joining in the variable student 1, we assign the value 2015. Observe that year of joining itself does not have any

meaning unless you say in which variable the value year of joining has to be modified. In this case, we say that student 1, in student 1 year of joining is assign the value 2015 and the dot delimiter is used to say in which variable the corresponding field.

We can also assign the variable student 1 to student 2, exactly as we assign variables of other basic data types which we have familiar with. Now, when you want to copy a string into the field name in the variable student 1, we use the string copy command. The string copy command is present in the string library, for this you need to include the header string dot h. Then you can like every other data type print the appropriate values that you want, here is a printf that does this.

Now, comes the interesting thing, so these four points were really about how to use a variable of a new data type that you have created. In this case, how does one use variables student 1 and student 2 of the data type struct student details that you have just created. And those are the things we have seen, we have seen that you can perform assignments and you can copy a string value into the appropriate character array, you can print the corresponding string.

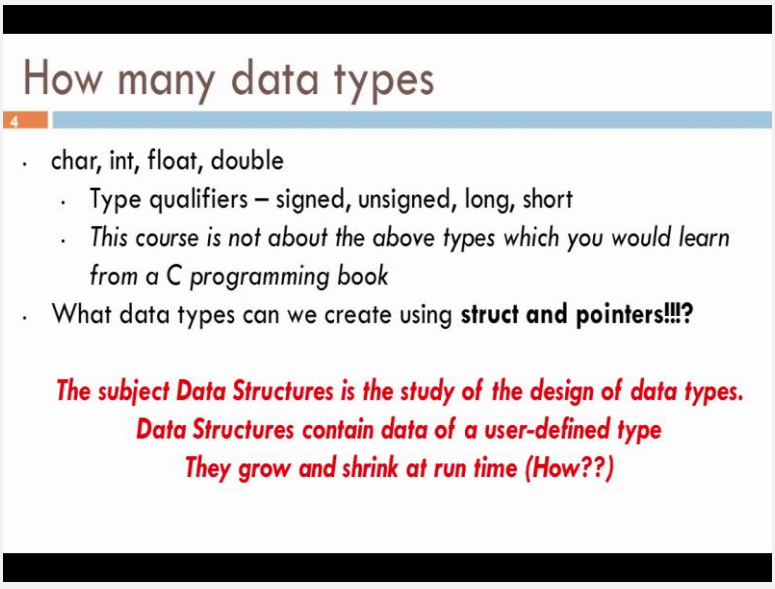
Therefore, all these assignment operations can be done; however, the thing there not written here, but I am sure you are aware of this, that you cannot perform comparison operations between student 1 and student 2. If you wanted to compare the contents of student 1 and student 2, you must compare field by field that is not written here, but I hope that was clear. Now, we come to the next concept in this slide which is the concept of a type name.

Recall, what we have done so far, we create a new data type called student details. Actually that is not correct, we have created a new data type called struct student details. Now, this is a very combustion or a very difficult thing to keep typing as a programmer, therefore a C programming language allows us to create a new type name and a new type name is called student record and the statement in red which is typedef, this is a key word, struct is a key word, student details is the name of the structure that we have defined in the previous slide, student record is a shortcut for struct student details, I hope this is clear.

Therefore, for a programmers comfort a new type name is created using the typedef construct, there are little more details about typedef, but however, as a programmer as a starting point it is useful for you to understand that you have created a new type name and you have set that the type name student record from now on stands for the new data type struct student details.

How do you use this to define variables? For example, I have given the most interesting definition, so I define a pointer to student record and the pointer is the variable student pointer. So, therefore, so far we have seen that using the struct key word, you can define a new data type, when you define a new data type you say, what are all the different data items which constitute that particular data type. Now, again you specified a basic data types there and then, from the programmers comfort you create a new type name. This is a very useful feature, and let us see how to use these in a subsequent slides. But, at this point of time you should be aware of the fact that you can now define new data types.

(Refer Slide Time: 12:49)



Now, let us just start for a minute and ask how many data types do we know now. So, well we know characters, we know integers, we know floating point data types, we know double procession and many of these also have type qualifiers. For example, you must have seen signed integers, unsigned integers, sign characters, unsigned characters,

longed and short type qualifiers, these are qualifiers that are associated with the data types, character, integer, float and double.

This course is not about the data types, but you can definitely learn about these from a C programming book by taking appropriate applications and definitely we will design appropriate assignments, so that you will get to use all these features. So, you may have to enter into a steep learning curve to be able to program. Now, the question is what are the data types that we can create using struct and pointers, even more why do we need pointers?

So, observe the pointers are also a data type, an address is a data type and though we will not delve into this. So, whenever you define a variable to be a pointer, so the pointer data type, the pointer to an integer, pointer to a character and in the previous slide you would have seen pointer to student record, so the pointer to different data types by itself is a data type. I repeat, the pointer to a data type is by itself a data type by which you can create variables of that particular data type.

Example, observe that student pointer is a variable and if somebody asks what is the data type associated with this variable, the answer that you have to give is that the data type of this particular variables student pointer is a pointer to student record. I repeat the data type of the variables student pointer is a pointer to student record. Therefore, the pointer to a data type is also a data type. You will become familiar with this concept as you go through this course, but I hope if you have a confusion, go back to what I have said earlier and it should become clearer.

So, therefore what is the subject data structures, many people have post this question on the forum and I am answering this question now. The subject data structures this is the study of the design of user defined data types. I hope this is now clear, if somebody says he or she is studying data structures, the picture that has to come to your mind is that this person is designing new data types and using those new data types in different programs.

Now, what is physically a data structure? Observe that the previous statement was about the subject data structures that use study. Now, you can ask what is a data structure, to

define a data structure I will say what a data structures contains, a data structures contains data. Now, what kind of data, data of different types and in particular I will say data of user defined data type. Now, this is not completely correct, because an integer and character and float and double are not user defined types.

But, in this course when you study data structures, you will be creating your own types. Therefore, we will be creating data structures which contain data of types created by you or types created by me. Therefore, you can now visualize a data structure to be something that contains data of a certain type. Now, what are important properties of data structures? So, the first part is very clear, a data structures contains data.

Now, if you ask me data of what type, I will tell you look the most important thing is user defined data type. Then you can ask me, is it only user defined data type, then I tell you no, no you can also have data of integer type and float types and so on which are provided by the programming language. So, therefore data structures contain data of different types, the subject becomes very interesting when you start trying to create your own data types.

What more do we know about the data structures itself. We use our data structures to be dynamic, in a sense that we want to write our programs, so that the data structures can grow and shrink at run time. That is as your programs runs, your program must be flexible enough to be able to maintain data which grows in volume for a certain amount of time that is it is processing more and more fields, more and more data items and after some time as the processing gets over, it also shrinks periodically and we will see applications.

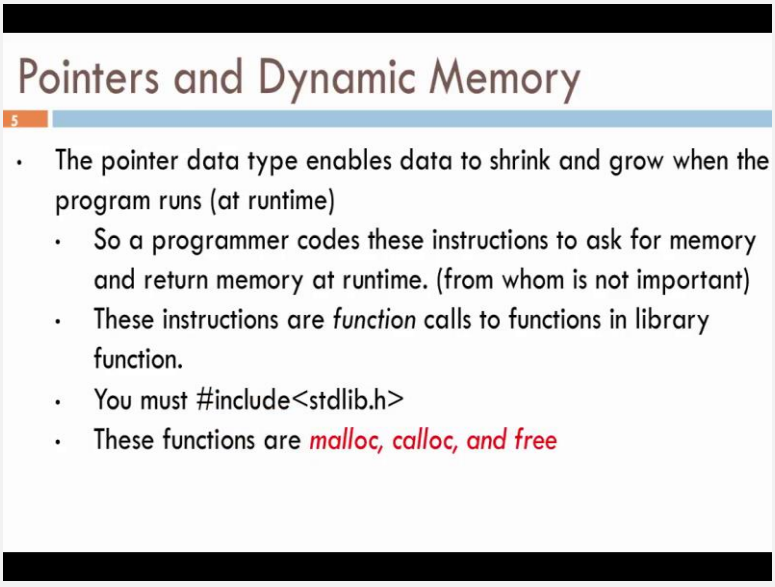
Now, you will ask how does one writes such programs? That is the focus of this course. I hope this is now clear, the summary of what we have discussed so far is that we know that the struct key word in the C programming language allows us to define a new data type. The typedef key word allows us to define new type names as a shortcut for programmers. We also know now that there are many data types that we know and for the rest of this course, the most important data types that we will repeatedly encounter are user defined data types created using struct and very importantly something that I

have said I think four times and I am going to say this fifth time that pointer to a data type is a data type by itself.

In other words, pointers are data types, address are data types. Now, using the freedom to create new data types, we will now write programs which create and maintain data structures. In other words, we will write programs where the data of a particular type that we have maintaining can grow and shrink at run time and we are going to see how to achieve this.

If a data structure has to grow and shrink at run time, somehow we should be able to get memory from somewhere, whenever we need it or whenever the execution needs it, we should be able to put an instructions, so that at run time the system will, the program will not only execute your arithmetic and logic instructions and an assignment instructions, but we need to also put an instructions which will demand memory from the underline execution environment. It may be the operating system or whatever, it does not matter you can think of this is an abstract person, who is giving memory as long as it is available to your program, when the program is executing.

(Refer Slide Time: 20:01)



So, now pointers play an extremely important role in enabling data structures that shrink

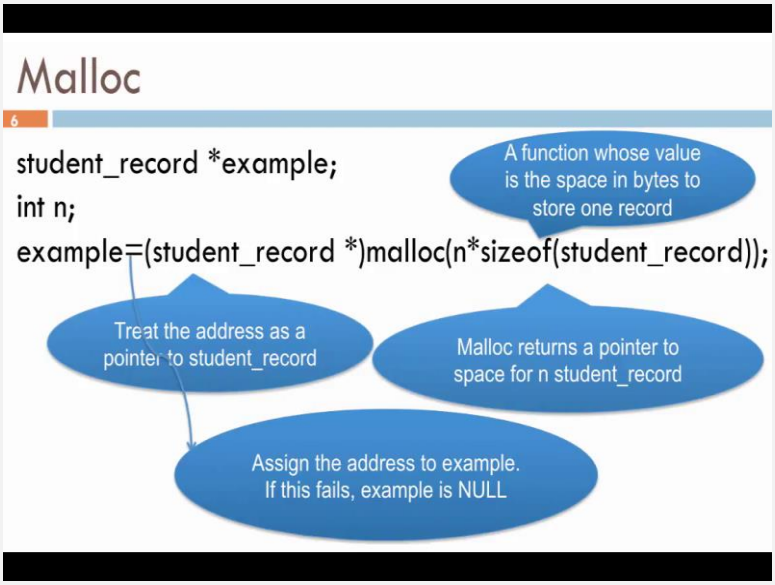
and grow when the program runs; otherwise, called run time. Like I said little earlier a programmer codes it is instruction to ask for memory and it is also important to return memory at run time from places when it is not important. So, for example, if you have finished using a certain amount of memory that was dynamically allocated to you, it is a good programming practice to return that memory using appropriate instructions when you program.

Now, you can ask what are these instructions? So, are these instructions keywords or these functions calls, the answer is these instructions are functions calls, these are functions which are there in the C library and for these you must include a stdlib dot h. In all these I am assuming that you are programming in a Unix or Linux like environment. However, if you are programming in other programming environments, other underline operating system you may have to find out, what needs to be included and you can post such questions on the forum and we will see if it become answer.

So, you need to make library function calls and what is the result of this function calls, you can ask for a certain amount of memory and you can also release the memory that you have finished using. And remember that these are function calls that you are write into your program when you compile and run, these functions will be executed. The result of this function calls is that you will get memory and you can return memory.

Now, three of the functions that we will study are called malloc; otherwise, called m alloc and free, these are the two functions that we will study in the lectures. Calloc is another function that you can study by yourself which is called contiguous alloc, calloc, but we will not really talk about it. Malloc or calloc and free that is what we have going to talk about and these are really the functions that give our programs the power to grow and shrink to maintain data structures that grow and shrink at run time.

(Refer Slide Time: 22:26)



So, let us look at malloc and this slide is a heavy slide and you may have to go through the couple of times. But, it is a complete slide it has everything that you need know about how to use malloc. So, let us just see student record is the data type do not forget that this is a new type name we created in the first slide. Example is a variable and just to ensure that you do not forget, example is a variable of what type, if you try to answer it yourself, but the answer is that example is a variable of pointer data type, pointer 2 what pointer to student record data type.

So, example is variable that contains the address of a location, which contains data items of the student record data type, I hope this is clearly. Then n is an integer variable they should not be confusing at all and way we are going to use this is that we have going to use n as the number of data items of student record type that we want. So, I am going to use n as measure of size. Size of what? It is going to be number of data items a student record type that I want to ask, I am going to ask for that much amount of memory I am going to ask for n units of memory in the next statement.

So, let us just look at the statement, the statement reads it is an assignment statement, on the left hand side of the assignment statement is a point a variable. Example, it is suppose to store an address and the right hand side is a function evaluation. So, now let

us evaluate the function from inner most function called to the outer most evaluation. So, the inner most is this function called size of student record, size of is a library function.

And what is the input argument to this function? The input argument to this function is a type name. So, for a example you can write a small program which says size of int or size of care or size of float. So, therefore, the argument is a type name, size of is the name of function, the value is the space in bytes to store one data item. Now, I am not going to tell you how much space student record is going to take, you can calculated, but that will not be the exact numbers this various from compiler implementation to compiler implementation.

But, you can get an approximate understanding, if you have data type has one integer field alone. So, let says it has two integers fields, then you know that student record must have space to store two integers. So, you are write; however, for the variety of other reasons, student records is often slightly larger than the estimate that we will make by adding of the amount of space required for the each of the individual fields.

And the reason has you do with the subject of computer organization, how memory is access and so on and so forth. So, we will not delve into exactly what the space use by student record has… However, size of is library function that takes a data type name as an argument and the value that returns is an integer, a positive integer value and unsigned integer value, which is the number of bytes required to store this one record.

And now I have an expression it said, n multiplied by the result of this size of. So, and demanding using the malloc function, I am asking for n units of the space required for one student record. In other words, this malloc n multiplied by size of student record given as the argument, essentially asks the system for n units of space, each unit being the space to store one student record.

Now, what is malloc to malloc returns a pointer, in other words malloc returns an address, you can imagine that malloc goes an ask as systems saying, give me space for n student records and the system gives returns a pointer, it says if you go to this address that many bytes are a essentially space that are given to you that is return to you. Now,

because it is a pointer, it is a safe programming practice it is not necessary. But, this operation, this expression which precedes the function call malloc that is in brackets you return should an records start is called a typecasting operation.

So, what is the typecasting operation, as a programmer you say that whatever memory I get successfully from malloc should be consider as an address to student record data type. So, I repeat this the pointer that malloc returns is an address to a certain amount of space and the amount is space that is given to use the space for m student records. As a programmer you also specify explicitly that this space must be treated us m student records.

In the C programming language this is really not necessary by default the address will be consider as a pointer to student record. However, it also allows you to do some tricky programming here, which I will not draw your attention to at the moment. So, therefore, the student record star, this is type casting operator which precedes this particular function called is not mandatory. But, it something that I am exposing you therefore, this pointer is now assign to this variable example.

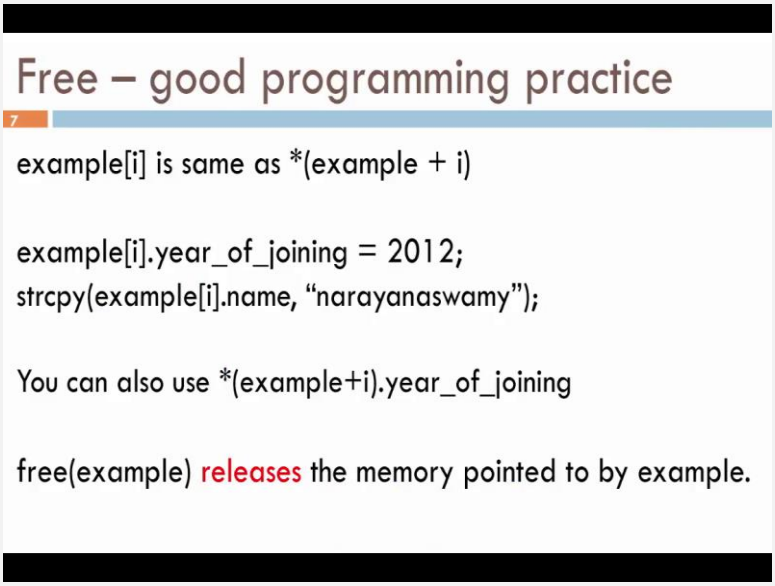
Therefore, you can see that there is no type mismatch in this statement that is the data type on the left hand side and the data type on the right hand side or identical. Sometimes, because your systems has run out of memory malloc may fail, in which case example will take a constant value called null and you can check if example is equal to null before you use the pointer that is good programming practice. So, what are we have done, so for let us stop and takes for.

So, we have talk about struct creating a new data type, we define the new type name, we define the new type name, we looked at what data structures are. And finally, we convince it is our self's that being able to manipulate memory to be able to ask for memory and to be able to play with addresses very important for creating data structures and maintaining and managing data structures, if you do not do it somehow your program has to do it this.

So, in the C programming language all the details are exposed to you, in other rich

programming languages many of these details may not be exposed you. For example, if you go to java the details of the addresses and so on may not be exposed to you, to the programmer. Whereas, in the C programming language and this what I mean it is an very flexible and nice programming language, you have access to all the internals. Of course, you have to choose to program very carefully when we you have access to many features. So, now we exposed ourselves is to how malloc works or rather what malloc does and what kind of statements you will typically write involving the malloc library function call.

(Refer Slide Time: 30:01)



So, let us just look at some programming practice's, recall in the previous slide we define example as an address. So, now I can say example of i observe that this is like and array access. Now, what is example of i? If you should answer this question, in other words when somebody asks what is example of i the question typically means, what is the data type of example of i. So, what are the alternatives you may say well, it is of data type student record, some of you are not very comfortable may say it is pointer to student record.

The second answer is wrong, the first answer is right, example of i if you imagine, the data type of example of i is one student record, not a pointer it is one student record. It is

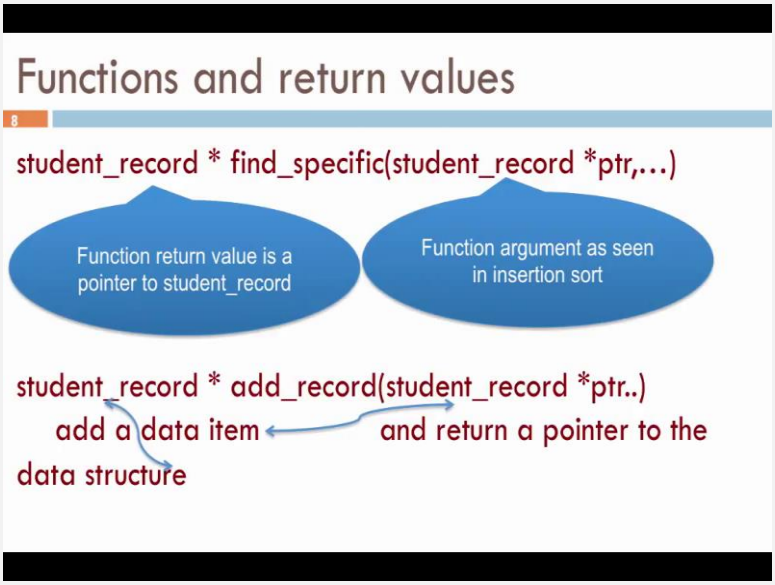
same us example which is a pointer plus i this plus i says, whatever example is pointing to increase it by i units, I repeat increase the by i units. How much is one unit? In this case one unit is this size of your student record. So, increased by units and look at the data item that addresses pointing tool.

We have to program with this to be able to feel more comfortable and I encourage you to do this, then I can say example of i point dot year of joining is 2012. So, this is the assignment that we saw in the first or the second slide. Similarly, I can say copy the shrink narayanaswamy into example of i dot name. Now, you can also use star example plus i dot year of joining, in latest slides you will also see some another shortcut, which is equivalent of course, I do not explicitly mention at I will use it. So, you should figure out what it is of course, I will talk about it.

Now, the function call free example releases the memory pointer to by example. In other words, after you done a free, example is not pointing to anything meaningful. And somehow the underline execution environment gathers all thus memory locations that you have released. And that memory can be given to other function calls that you are making or to other processes that may be running on your system and so on and so forth.

Therefore, freeing is a good programming practice, on the other hand if you do not free then you may end up using a large amount of memory and at times you want running out of memory. Therefore, as a programmer it is good for you to analyze and imagine how memory is been utilize and release memory back to the underline execution environment, whenever appropriate for your program.

(Refer Slide Time: 34:10)



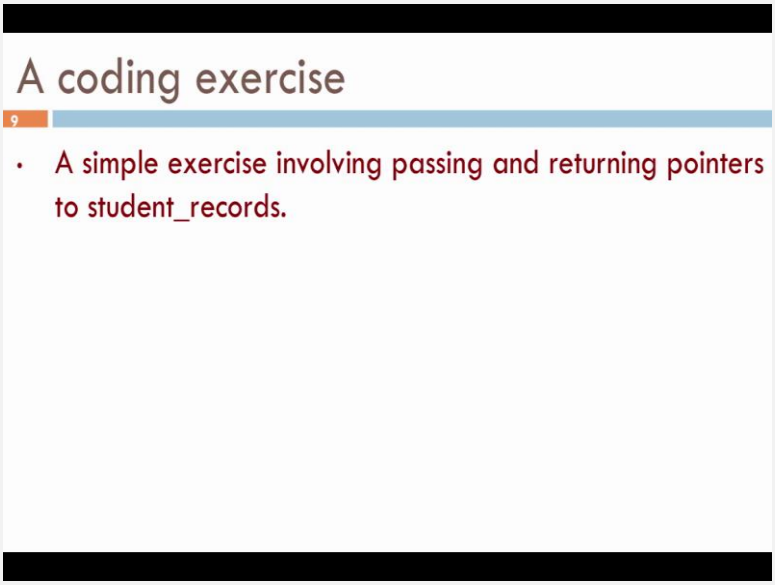
Here, comes the third very important thing, how does one use all these things in function calls and how does one return values and this is extremely important. So, let us look at a generic function, it is find a specific item and return a pointer or returns it location, this is the generic function call. I repeat this again, when you are working with data structures one of the most common things that you will repeatedly do is you write a function that looks for a certain data item in a certain place and you will return a pointer.

And this is exactly what is illustrated and this example, find specific is the name of the function. What is the argument? The argument is an address which points to student record. So, you can imagine that the pointer points to an array of type student record, so I repeat the pointer points to an array of type student record just imagine it. So, when we write the program you will see this a completely realized. And the result of the find specific function is that it return the pointer to student record.

What is the another very common data structures function, that you will repeatedly write, you will say add a record to this particular data structure that is which particular data structure, the data structure pointed to by pointer and the particular record and return a new pointer. So, that is a typical data structures function, similarly you can say delete record and so on and so forth.

So, you will find the particular pattern or find the particular value, you will add a particular record, delete a particular record. And after performing these operations, you will return a pointer to the modified object. So, this is the template function called whenever we work with data structures.

(Refer Slide Time: 36:32)



So, let us just take start we study three things, so for we talked about structures, we talked about type names that is the first concept, then we convinced ourselves that a data structures is something that grows and shrink as the program execution continues as the program executes. Therefore, as the programmer you should put in appropriate instructions or appropriate statements to grow and shrink your data structures that you have imagined.

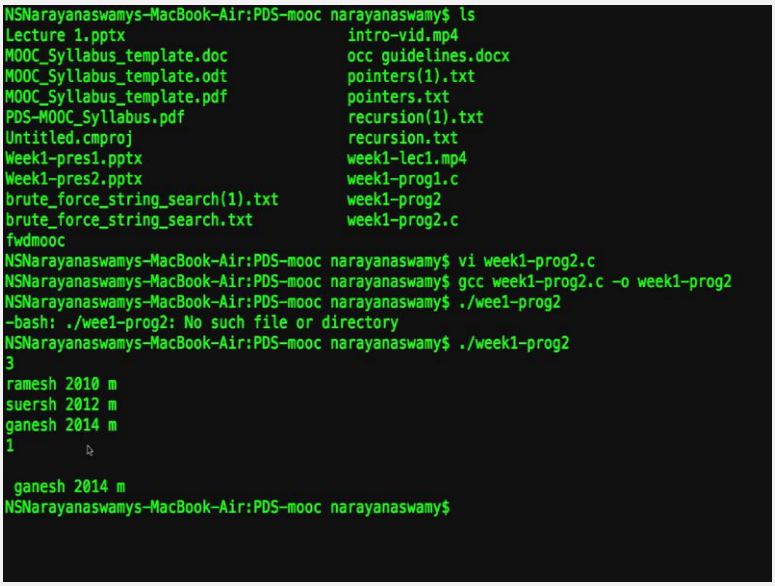
And the next point is that... So, the next point which is important is that you need to find out what statements or relevant for you to obtain memory from an underline execution environment and we have seen that the library std lib dot h, the header file provides you access to malloc and free functions. And we have seen how to use the malloc function call and all the parameter is associated with it in a careful way, we will see more details of it in the program that is going to follow.

And finally, we also saw what at typical function call looks like, when you work with data structures. So, a typical function call looks like finding a particular key and returning a pointer and in a particular record and returning the pointer, deleting a particular record and returning a pointer. So, this is essentially how the whole data structure area works.

So, you use certain features is say this is a new data type, which is interest for me I want to give this shortcut name to this, I want to instantiate this particular data type I want to have, so much memory while execution for two represents this data structure. And you also have appropriate methods as they are called function calls to modify your data structure. And this is how the whole data structure evolves and this is how you program to create and maintain data structures.

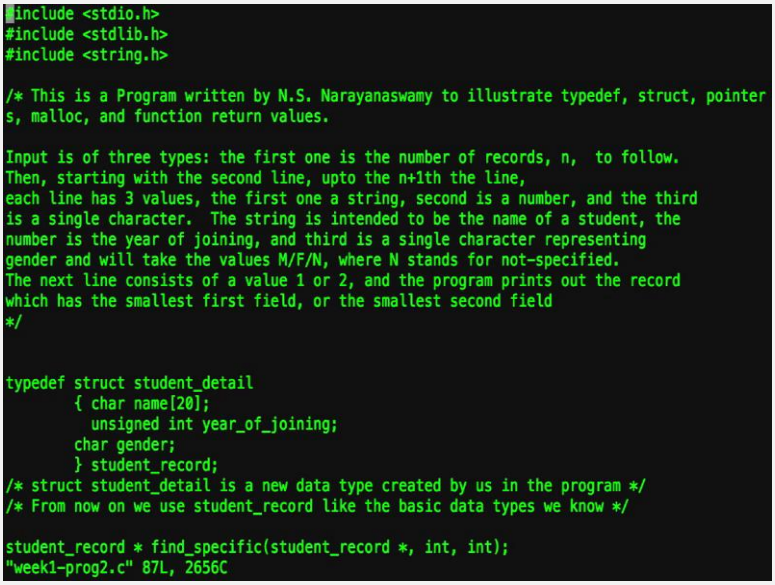
And every data structures has an associated data type and data type is something that you design and create. So, this is that was a summary of today's lecture and let us now look at this programming exercise like the previous lecture.

(Refer Slide Time: 39:07)



So, now the program is call prog 2 dot c week 1 prog 2 dot c that is not, so important.

(Refer Slide Time: 39:19)



Now, as you can see I have three headers which are included stdio dot h that is first standard input output, print f, scan f etcetera are presents in this header file, stdlib dot h and this is where malloc is presents, the functions call malloc is declared inside stdlib dot h, string dot h has the string operations that we are going to perform. So, we have move to the second part of this lecture.

So, it might be instruct to for you take a break and come back and look at this C program, though I am continue in this as one single record. So, what is this programming exercise, as in the previous lecture the intent of this particular program has very suck singly put on this the program written by me to illustrate the concepts of type def, struct, pointers malloc and function return values. In this example you will see, all these five concepts illustrated at one place.

Let us understand the problem statement, the problem statement is the following. The input to this program as have three types, the first input is an integer value n which tells you how many records are going to follow, then starting with a second line up to the n plus 1th line, each lines has three values, these values are separated by a space. The first one is the string, you can imagine in a string to be the name of a student, the second one is a number you can imagine that to be the year of joining of a student and the third one

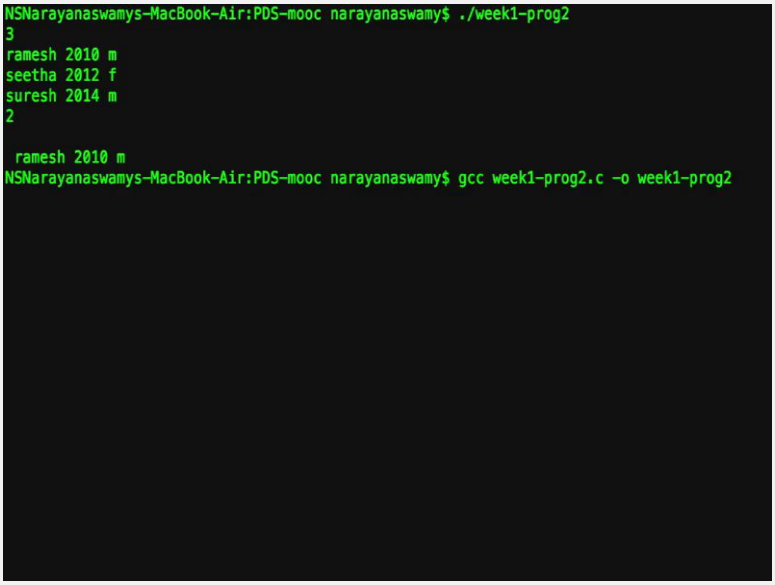
is a single character and it will represent a gender which can be M, F or N we stands for not specify.

So, subsequently the last line, so observe this the first type told you how many records, then the number of records followed and the last line consists of a value 1 or 2. What is the meaning of the value 1 or 2? The value 1 or 2 determines the output of the program, depending on 1 or 2 the program prints out the record which are input the smallest first field or the smallest second field, the programs prints out that record among the n records that you have input.

The first one that has the smallest first field or the smallest second field. So, this is the program let us just do and illustration before we go on to the program ((Refer Time: 42:29)). So, already have complied, but so let me just run a compilation again using the gcc compiler. The compilation has successfully completed without any messages, now this is my executable that I just created, let me runs. Let us say that there are three records, now I have to input the record, let say Ramesh 2010 and Suresh 2012 and Ganesh 2014.

Now, we want the record which has the lexicographically or rather in the dictionary order the smallest first field, in other words I want that record which has the name which occurs first in a dictionary order. So, let me give 1, as you can see this is the output of the program which is Ganesh 2014 and that is very clear Ganseh is lexicographically smaller then Ramesh, which is lexicographically smaller then Suresh in the English dictionary order, let us run one more run of this program.

(Refer Slide Time: 44:03)

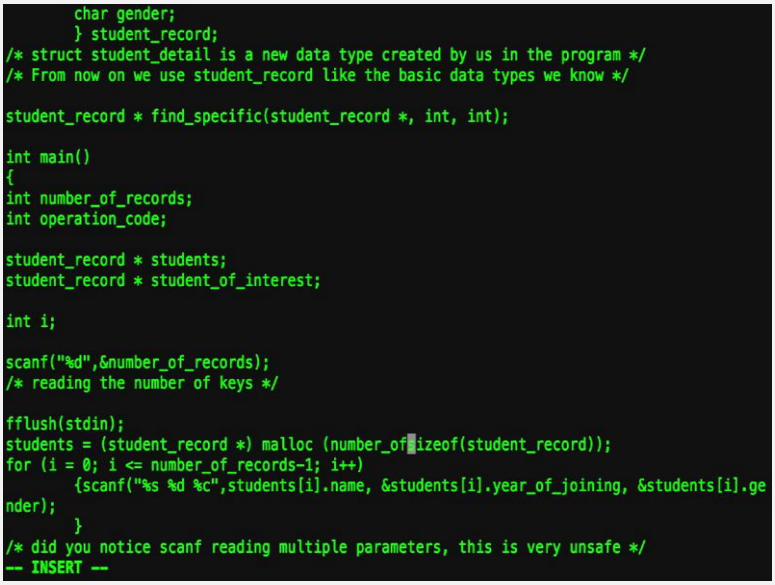


Let us say again there are 3 fields is Ramesh 2010 male, Seetha 2012 female, Suresh 2014 male. And let us say I want that record which has the smallest second filed, in other words you want to student who join the earliest. As you can see the output is the person who joined in 2010. Now, clearly you can immediately imagine multiple applications of this programming exercise I do not have to tell you, if you have use any software.

For example, if you use an excel spread sheet, this is a standard feature where you have some data which occurs and columns and then you ask to, short according to the first column or a second column or a third column. So, if have not done this please go back and try this use Microsoft excel or there is also open office, which also provides similar features go ahead and use it, create a few columns of data in first column you can put the names of your friends, the year of birth of your friends and the gender of your friends and you can ask the spread sheet program or the spread sheet software, the sort based on the first column or base on the second column and so on.

So, this small programming exercise that I have written in C is something that illustrate such programming exercises. So, you see the input output behavior of the program I hope the input output behavior is clear, let us go back to the program itself. Let us look at the new data type that we have created.

(Refer Slide Time: 46:11)



Here as suppose to what I have done in the lecture? I have said that I want to create a new type name calls student record, type def the consequence is new type name calls student record, the student record type name is a shortcut for struct student detail. Let me say this again, type def results in a type name calls student record, student record is a shortcut for struct student detail.

And what is struct do struct says that, struct student detail is a new data type, whose fields are name with 20 characters, year of joining is an unsigned integer as a programmer you are explicitly stating that this is an unsigned integer. And here giving the variable call gender, you have variable call gender which stores a single character. So, therefore, this is the new thing we are going to create a new data structure, the elements in this data structure are going to be of data type student record or going to have type name student record.

Student is the same type as struct student detail that is student record is the shortcut for the type struct student detail. Every item of this particular data type has three fields which are name, year of joining and gender, the first one is a character array, second one is an unsigned integer and third one is a character. So, this is the first thing that you do when you want to write a program you understand what kind of data you are going to

manipulate arrange them into appropriate data types, you create a data type give appropriate data type names.

So, that you can comfortably to use them that is the first step and we have done it. The next step is to like in the previous lecture, define this function prototype which says that find specific is a function name, which is going to take a pointer to student record one integer and another integer. We will see what these to integer argument stands for and it returns a pointer to student record,. So, now, you defined your function prototype also.

So, let us quickly go through the things at you would familiar with by now, this is the main function it returns an integer value, it has one integer variable call number of records, this is the variable which gets it value from the input, operation code this is the code that you keyed the value 1 or 2 that you require to identify the record, whether you need to find the record which has the smallest name or the record which has the smallest year of joining that is stored inside operation code.

When you say the student is a variable of type pointer to student record. I repeat student is a variable of type pointer to student record. Similarly, student of interest is a variable of type pointer to student record, then I have int i is a temporary variable. So, this scanf is what is do, it reads the number of records, the first reading from the key, you can ignore this fflush stdin for now we will talk about it a little later. Then comes the third major concept of the day.

How to use malloc? I ask the system at run time to allocate memory, which is size of student record. So, therefore... So, let me there is a missing detail here I am going to make this change right in front few here as a record this must be number of records star size of student records. Now, it is the interesting it still work without this, so let us understand what we are doing, I am going to ask for memory which is number of copies, number of records of student record, like we saw in the slides.

And then this is the type casting and students is going to contain be address of the memory region that has been allocated to him. So, what does malloc do malloc preserves a certain amount of memory for your program, for your execution and returns a pointer

to the first of those memory locations and students contains that particular address because of this assignment, I hope this is clear. Now, what we do the go to the input and read one after the other a string which corresponds to the students name and integer which corresponds to the year of joining and a character which corresponds to the gender.

(Refer Slide Time: 52:31)

```
int i;

scanf("%d",&number_of_records);
/* reading the number of keys */

fflush(stdin);
students = (student_record *) malloc (number_of_records*sizeof(student_record));
for (i = 0; i <= number_of_records-1; i++)
        {scanf("%s %d %c",students[i].name, &students[i].year_of_joining, &students[i].ge
nder);
        }
/* did you notice scanf reading multiple parameters, this is very unsafe */

/* The records are in the students array now */
scanf("%d", &operation_code);
if ((operation_code != 1) && (operation_code != 2)) return -1;
/* if operation is invalid exit with a return value that can be used by a calling program
 */
student_of_interest = find_specific(students,operation_code,number_of_records);

/* student_of_interest has the student with the least value in either field 1 or field 2,
as asked by operation_code */

printf("\n %s %d %c",student_of_interest->name, student_of_interest->year_of_joining,stud
ent_of_interest->gender);

printf("\n");
```

And observe I am using a single scanf which is reading three parameters, separated by a single space. I repeat this it is this scanf statement reads 3 data items, one is a string he goes into name, the next one is an integer it goes into year of joining and the last one is a character which goes into gender. And each of which goes into the ith record in the data that in the memory that has been allocated to this execution by the system, it goes to the ith record, the ith record gets student record in the memory that has been allocated, because of this malloc function call that is what that loops does.

Next what we do is we read the value for the operation code, the next sentence or the next statement is essentially says that if the operation code is not one or not two return a minus 1. Essentially the operation code is invalid, no shorting, no operation can basically we done, you return a minus 1 that is the program exits by returning a minus 1 to whom is not important for now, it does return a minus 1 we will worry about the value minus 1

is return to whom a little later are as you program more and more you will become more comfortable with this.

And especially if you work an unique programming environment, this becomes very clear. Now, comes out new the function call find specific which we make, we sent the students pointer to the function, we send the operation code which is either 1 or 2 and we also give the information of the number of records that we have in the students array. I am going to call the students pointer now also the student array inside this loop, they are equivalent there is one in the same.

And the return value that is, this is the pointer student of interest is a pointer and it points to the record in the students array which has the least value of field 1 or field 2 has given by operation code that is what is put down in this record. And this printf statements finally, the printf statement below prints the whole, thing prints the three data items which is what we have seen in the program, it gives a slash n before printing and gives a slash and after printing, so that you can see this clear.

(Refer Slide Time: 57:49)

```
fflush(stdin);
students = (student_record *) malloc (number_of_records*sizeof(student_record));
for (i = 0; i <= number_of_records-1; i++)
        {scanf("%s %d %c",students[i].name, &students[i].year_of_joining, &students[i].ge
nder);
        }
/* did you notice scanf reading multiple parameters, this is very unsafe */

/* The records are in the students array now */
scanf("%d", &operation_code);
if ((operation_code != 1) && (operation_code != 2)) return -1;
/* if operation is invalid exit with a return value that can be used by a calling program
 */
student_of_interest = find_specific(students,operation_code,number_of_records);

/* student_of_interest has the student with the least value in either field 1 or field 2,
as asked by operation_code */

printf("\n %s %d %c",student_of_interest->name, student_of_interest->year_of_joining,stud
ent_of_interest->gender);

printf("\n");


return 0;
}
```

So, let us just take stop what is the main function do, the main function has demand a certain amount of memory form the systems, after reading the numbers of records as

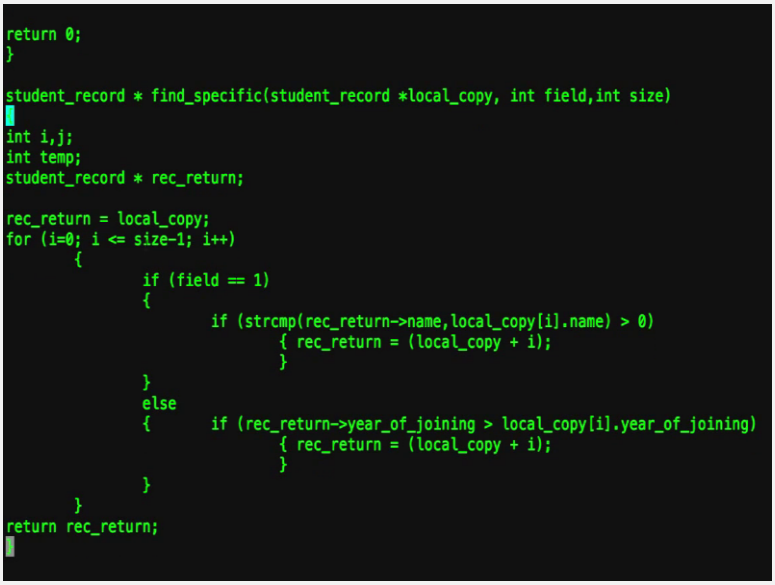
inputs, it us ask for the certain amount of memory. Then it has populated the memory with appropriate values by reading the input. Now, it takes the operation code, if the operation code is invalid, the program will exit if the operation code is valid, then control will be transfer to the find specific function.

Now, let us go to the find specific function, after the find specific function returns a pointer to the appropriate data item, this printf function prints the three values which are of interest, it is very simple program conceptually. But, it will illustrates all the different important concepts which are necessary for us to create data structures throughout this course.

The find specific function is very simple, the local copy the variable name as call local copy, it is a pointer to student record this is the field that you are interested in arranging a data type by data by that is you want to look for the appropriate record, for which the field value is the smallest and this tells you how many records are there that you must process. And the return type is a pointer to student record, as we have been saying this again and again.

So, i and j a temporary variables actually j does not get use, temp is also not used here instead of temp I am using a new variable call rec return which is the pointer and this is the pointer to student record. So, this is the essentially the short form for the value that is going to be returns, this is a variable name, initially rec return a set to be the local copy that is you points to the first record in your data.

(Refer Slide Time: 57:49)



Now, let us just look at this one, so here is a loop we checks each of the elements, if the field is one it does not string compare of the string which is available or the name which is available in your rec return and the string that is available in local copy of i in compares two of them that is strcmp compares these two strings. If the first argument is the lexicographically larger than the second one, then the return value 0 is greater than 0 actually it is 1. And is the first argument is the lexicographically smaller then the return value a minus 1, if they are equal then the return value 0.

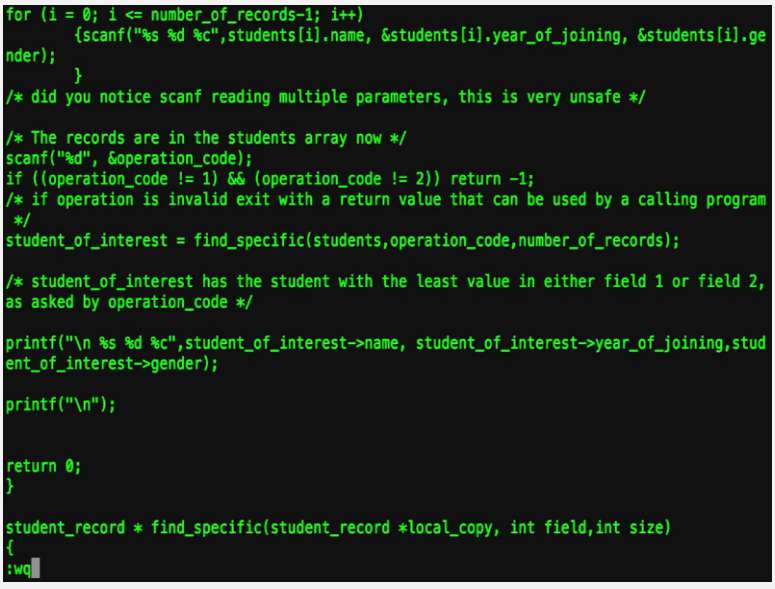
Let me just repeat strcmp is a library function in string library it takes two arguments, this is the first argument it is a string, this is the second argument which is the another string it compares the two of them, if they are equal the return value 0, if a first argument is smaller than then second argument, the return value is minus 1. And if the first argument is larger than the second argument, then the return value plus 1.

So, if the name in the current smallest which is your candidate is larger than the name in the ith local copy, then you say that rec return should contain the address of the ith local copy which is what is here. So, observe there is not type mismatch here, this is the pointer, local copy is an address, incrementing an address by a i units is also an address. Therefore, there is no type mismatch in this statement, I hope this is clear.

This is then if the field is one that is you want to do this whole selection according to the first record, the first field in the input records. On the other hand, if you want to do the selection base on the second field that is just an integer comparison, you compare the year of joining with the year of joining that is you compare, your current candidate year of joining which you believe this is smallest, which you have stored with the year of joining in the local copy of i that is in the ith record.

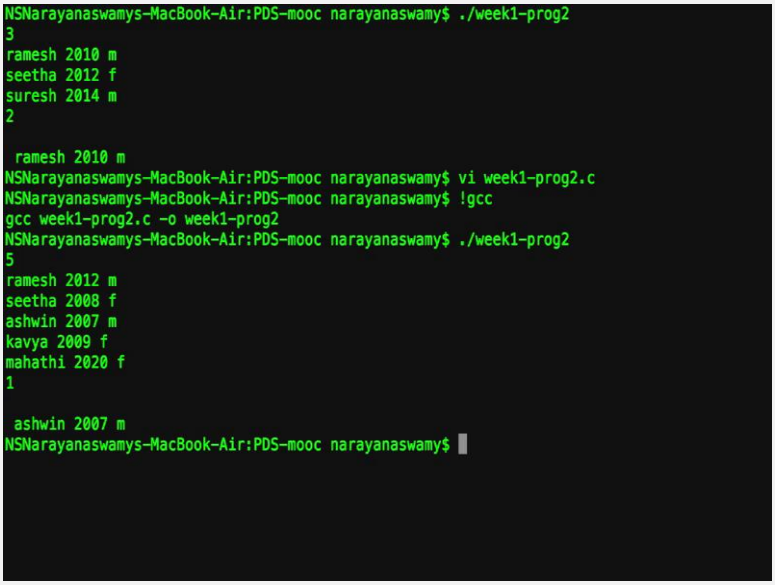
If the current year of joining is larger then you update the appropriate record address. So, which is something that you have already seen in the previous lists. So, observe that the field is use to perform two different types of comparisons, here you perform a comparison of two strings. Here you perform a comparison of two unsigned integers, after now processing this whole sizes numbers of records definitely you would a form, the record with the smallest appropriate field and this value is return into the main function and then subsequently the value is printed.

(Refer Slide Time: 61:01)



So, this completes the whole description of the code, and just because we made one compilation change.

(Refer Slide Time: 61:10)



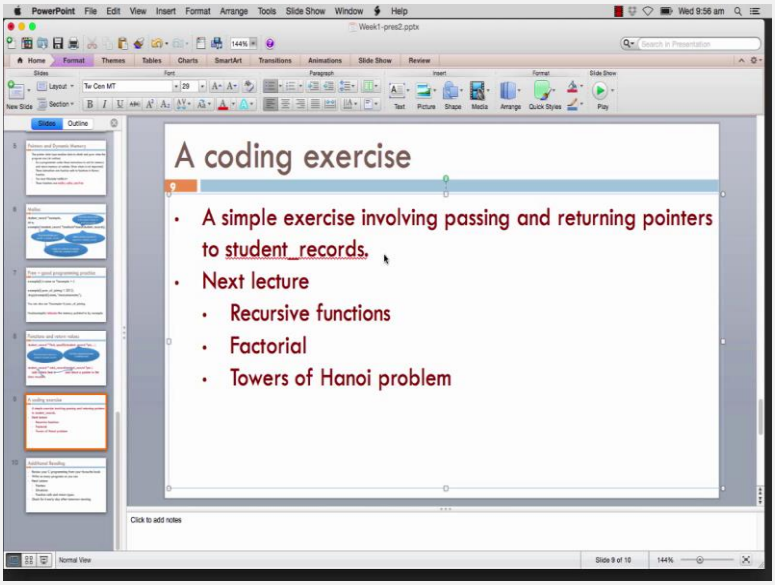
Let us just complete the compilation goes through let us executed once, let us say they are 5 records Ramesh 2012 male, Seetha 2008 female, Aswin 2017 male, Kaviya 2009 female, Mahathi 2020 female. So, now we have input 5 records and let us try to sorted according to the first field and get the smallest value; obviously, we have Aswin. So, let us also possible to test this program like in the last lecture by giving aeronauts inputs and see how the behavior is.

But, in the interest of time for this recording I will not do it, you welcome to write your own program and perform such test to mainly understand that it is not possible to control the input that is given to a program. So, it is very important to program in such a way that as much as possible you can capture meaningless inputs. So, we are almost at the tail of today class. So, let us just look at what we have done? We have studied three important concepts with respect to the subject to data structures.

The goal of data structures is to write programs, were the instructions of the statements enable data structures to shrink and grow based on the input environment that is the set of inputs that are given to the program while execution. For this we convince ourselves that being able to ask the system for memory at run time is a very important feature and we saw there are library functions which do this.

We also saw the data that comes to ask when you program maybe of different types and therefore we constructed this example of a student record data type. And we saw how to define a data type and then you also saw how to give a shortcut to this data type name. Finally, we saw the simple exercise which broad in all these concepts together, which picked up the least value satisfying the certain property from a input set of records of a particular data type. So, let us complete the lecture by going back to the....

(Refer Slide Time: 64:29)



So, therefore, this brings us to the end of this lecture, and in a next lecture we will work with recursive functions, specifically we will start off with simple recursive functions for factorial and we will also write recursive functions for the towers of Hanoi problem.

So, thank you very much and I hope you enjoyed the lecture today.