

Design and Pedagogy of The Introductory Programming Course
Prof. Abhiram G. Ranade
Department of Computer Science and Engineering
Indian Institute of Technology – Bombay

Lecture - 17

Advanced Programming Topics, 0: Introduction, Organization of Medium Sized Programs

Hello and welcome to the course on Design and Pedagogy of The Introductory Programming Course. The topic for this sequence of lectures is advanced programming. Let us first take stock of what we have done so far.

(Refer Slide Time: 00:40)

Let us take stock...

- ▶ Review of approaches to introductory programming.
- ▶ Design of an introductory programming course.
- ▶ Course pedagogy

Next: Some advanced programming topics

From the beginning of the course, we have reviewed approaches to introductory programming. Then we have designed an introductory programming course. We talked about the course pedagogy that is how to teach the course. In this, we considered mostly the core topics. So the topics, which we will use should be there in all versions, or whatever design you do for introductory programming, these topics should be there.

So these were the topics that we talked about so far. Today in this lecture, I want to talk about some advanced programming topics. Some of these should be covered in every course or at least some introduction should be given and we will see which ones, but most of these will be what we have been calling tier 2 core that is cover only if you have enough time or if somehow you

feel that your students would like to see some advanced material. So here is the outline of this sequence of lectures.

(Refer Slide Time: 01:46)

Outline

- ▶ Organization of medium sized programs Partially Core
 - ▶ Case Studies
- ▶ Advanced Memory management T2C
- ▶ Standard Library Partially Core
- ▶ Object Oriented programming T2C

In the first part, we are going to cover organization of medium-sized programs. How do we design medium-sized programs? So we will do this and this is partially core and we will do some number of case studies for this, then we will talk about advanced memory management. This is tier 2 core. We will talk about the standard library, part of this is core, part of this is tier 2 core, then we will talk about object oriented programming. This also is tier 2 core.

(Refer Slide Time: 02:27)

Organization of medium size programs

Most programs in intro course will be short, at most 20 lines.

Can be written "spontaneously", assuming we know the algorithm.

For 20 line programs there is little incentive for students to do "top down design" or "break up the program into functions".

We can try, but we risk sounding preachy.

To drive home the point about program organization we can try the following:

- ▶ Pose a somewhat large problem. Show a main program, with calls to unwritten functions, or incompletely defined classes/structures.
- ▶ The intent of the main program should be clear.
- ▶ Develop the rest in-class/ask students to develop as homework.
Many students like this if given in an assignment.
- ▶ After a few such assignments we can explicitly ask "isn't this a good way to organize code?"

The first advanced programming topic is organization of medium-sized programs. Most of the programs that your students will design in an introductory programming course will be short. They will be at most 20 lines typically. Now 20-line programs can be written spontaneously without really enormous amount of planning. We think about the algorithm. We think about how we do things manually and the sequence of operations that we have to do are not that complex.

Maybe there is a loop or something like that, but that is about it. So we are not really going to be concerned about how do we really organize this code beyond saying that it is going to contain a loop. So it will take maybe some amount of time to discover exactly what we are going to do or typically not even that possibly and then we will pretty much write the program without really too much planning.

Now if that is the case, if the program can be written in a spontaneous manner, then the design methodologies that are often talked about called “top down design” or the ideas which are suggested that break up the program into functions probably do not make that much sense or even if they do, it is hard to convince the students about that. Because they think that I can write this pretty much as I think and so why bother.

So we might try telling them during the course that look, you should do top down design or you should break up the program into small functions, but if there are only 20 lines or so that we typically write, then all these does not make sense and our students will think, oh, these people are just preaching, they are trying to tell us something, which is sort of supposedly good for you, but we should not really pay too much attention to them, because we can do without whatever they are telling us.

To drive home, the point about program organization, we need to do something else. So here is a possibility. We will pose a somewhat large problem. We will see examples soon and not only will we pose the problem, but we will show a main program as well. The main program will be short and it will contain calls to function, which have not yet been written and in addition, it may also contain incompletely defined classes or structures.

We are only giving some kind of a skeleton, but the skeleton is a proper main function. Furthermore, the main program has been written nicely in the sense that the names that we give in the main program are pretty self descriptive or maybe we explain exactly what different function calls are supposed to do. We are really giving the main program and we are also giving a high level view of what the code is doing.

Now the rest of the code has to be developed in the class. We could develop in the class asking students how do we do this, how do we do that or may be telling them, but we sort of walk it through with the students or we can ask the students to develop the rest of the code by giving it as a homework. The point here that the rest of the course is going to be small pieces and whatever skill the student has learned so far, will be enough to write those small pieces.

Now, it turns out, and this is my experience that if we do things in this manner, many students actually like it. After we have given one or two such assignments, we can explicitly ask in the class, “look we have been doing these 2 problems in this manner, so we had this main program and then we developed these function” and before we develop these functions, we just wrote calls to them, saying what we would have like the functions to do.

Before we declare the structure, we did not really know exactly what was going to be inside it. We had a rough notion of what entities the structures were going to represent. So this kind of high level planning is what is called top down design and is not this a good way of doing things. What I am saying is that we have to lead by examples. We have to give them a reasonably good example and maybe not just 1 example, but maybe 1 or 2 such examples.

After that you will see that we do not need to preach, students will want to do it in that way on their own. Let me take my first example.

(Refer Slide Time: 08:02)

Example 1: Assignment on Simulation of colliding balls

Problem introduction: Molecules of an ideal gas are often viewed as balls which bounce around in a box, colliding with each other and the walls of the container.

Write a program to: simulate a one dimensional version of this, and show its animation on the screen, starting from random ball positions and velocities.

Simulation algorithm:

1. Calculate times for collision between all pairs of balls, ignoring other balls.
2. Select pair whose collision time is earliest Will surely collide!
3. Move all balls forward in their trajectory till the calculated time of collision.
4. Simulate the collision of the selected pair: in one dimension, the collision simply causes the balls to exchange their velocities.
5. Repeat till required number of collisions have happened.

A main program using this algorithm is given next. You are to write the functions called by it, and define the structure Ball used by it. Some details about these are given later.

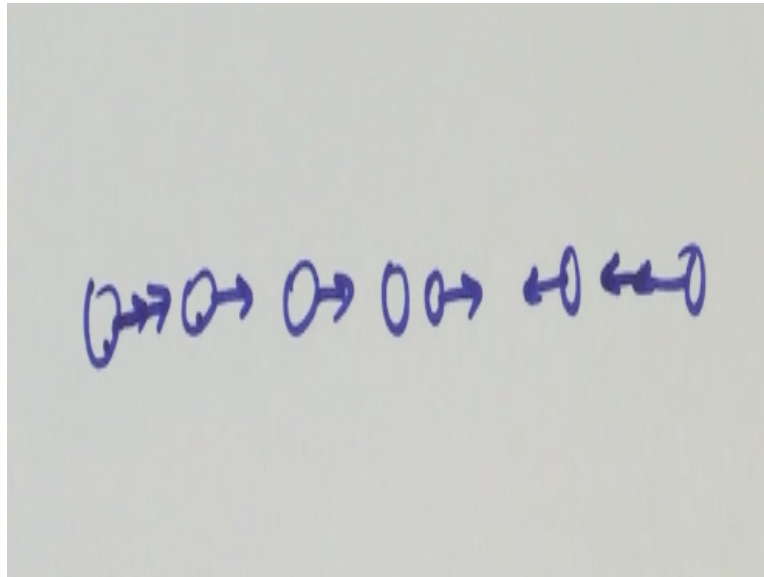
This is an assignment I gave on simulation of colliding balls. The problem was introduced and this is from the statement of the assignment as follows: Molecules of an ideal gas are often viewed as balls, which bounce around in a box colliding with each other and the walls of the container. You are to write a program to simulate a 1 dimensional version of this and show its animation on the screen starting from the random ball positions and velocities.

Now notice, that we have made 1 change. We have said that if you want to simulate an ideal gas, you really need movement in a box or movement in 3 dimensions. Now here, we are saying, we are only going to have momentum in 1 dimension. That is because if you have balls, which are moving in 3 dimensions, you are just going to manage a little bit more data and furthermore the way those balls are going to collide with each other is going to be requiring you somewhat more complex calculations.

The calculations and their complexity will involve some amount of trigonometry and in general some amount of mathematical complexity. The programming complexity will not be really high and since this is a course about programming and not about physics as such, but we just want to give the students an introduction or some kind of hint of what goes on physics, that is why we are giving this problem.

Since this problem is being given in a programming course, we should simplify the calculations, if possible and in this case, there is a perfectly nice way of doing that, which is we assume that the balls are moving around on a line itself. Maybe I will draw picture over here.

(Refer Slide Time: 10:08)



Let us say the balls are sitting on a line. They do not have to be equally spaced or something like that and the balls are may be moving. So these are their velocities. So the balls are moving, some balls are going to collide and then the process will continue. After their collision, their velocity will be changed and then the process will continue. Now exactly what has to happen or how do these collisions take place is a little bit tricky.

So we could have asked the students to design an algorithm. It is not really that tricky, but again because this is not a course in physics, we might as well tell them what algorithm is. The algorithm goes something like this. We first calculate the time for collisions between all pairs of balls ignoring other balls. For example, in this picture, we might say that look potentially this ball and this ball can collide, so that will happen if this velocity is larger than this velocity.

In this case, at what time they collide. Similarly, we can ask what about these 2 balls, what about these 2 balls, what about these 2 balls and so on. Now the tricky point is it seems just by looking at the picture that these 2 balls are going to collide first, but that cannot be true. If this velocity is

really high, then this ball may collide with this ball before this ball has a chance to collide with this ball. So it is a little tricky to decide exactly what happens next.

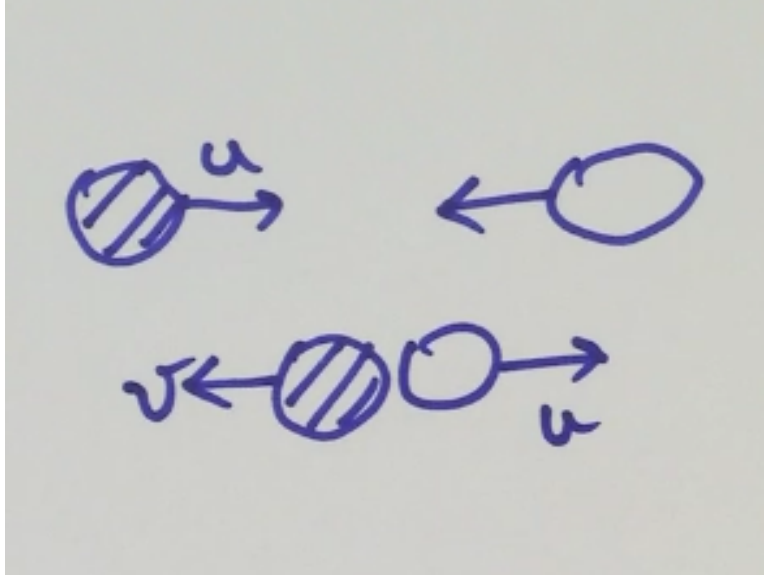
Which 2 balls collide first, but here is a simple resolution to this problem. We are going to select that pair of balls whose collision time is earliest. If we say, this ball and this ball are going to collide earliest and may be this collision happens after say some 5 seconds, then add every other collision is happening after 5 seconds and we know that this ball will have the chance to travel all the way till the collision and so will this ball.

Because other balls, which are colliding with these balls will collide only after 5 seconds if they get a chance to and therefore we can say that these 2 balls will definitely collide. So once we know which 2 balls definitely collide and the time at which they collide, then we are in business. We know that these balls will surely collide and now we are going to move all balls forward in their trajectory till the calculated time of collision.

So going back to this picture, this was the original position, then maybe we moved this ball over here, this ball over here, this ball over here, this ball comes over here, these 2 balls are going to collide, so they are going to be just about touching each other and this ball might have traveled until this point, something like that, some such motion will do. At this point, 2 balls, which we said are going to collide earliest will come and just about to touch.

They are just about at the point of collision. We are going to simulate that collision. So what happens in collisions. Again the physics here is very simple, which is why we are doing this one to mention. So let me show you.

(Refer Slide Time: 13:51)



In this case, if you have a ball and this has some velocity u and I have a ball over here, which has a velocity v and they approach and they collide, then after the collision, what is going to happen. This is some cross hatched ball and after the collision, this ball is going to have velocity v and this ball is going to have velocity u . So that is what the physics tells us. The physics in this case is very simple. The balls come together and then they essentially exchange the velocities.

You might think that well, that seems like what a coincidence, yes, it is nice coincidence and the physics causes that coincidence. That is convenient as far as programming is concerned. We simulate the collision and we exchange the velocities and then we repeat the whole thing again. If we said that we want this to happen for some 1000 collisions or whatever, I mean you might have said that we want this to happen for some amount of time.

Whatever it is, we can repeat the whole thing again. So this is the basic iteration which we are going to repeat. Now in the assignment, we gave the main program, which is going to use this algorithm and students were asked to write functions called by it and also define the structure ball, which is going to be used by the main program, exactly what those functions are, thing like that will be described after the code. So here is our code. This is the main program.

(Refer Slide Time: 15:36)


```

int main(){
    initCanvas("Collisions",1000,400);
    int n; cin >> n;        // number of balls
    const double r=10;     // radius of each ball
    Ball b[n];
    initializeBalls(b,n,r);

    repeat(100){
        double cTime;
        int leftCollider;
        firstCollision(b,n,r, cTime,leftCollider);
        moveBalls(b,n,cTime);
        setNewVelocity(b,n,leftCollider);
    }
    getClick();
}

```

We are going to begin by creating our canvas, so we are going to give it the title collisions, which will appear on top of the canvas and its width will be 1000 pixels, height will be 400 pixels. We could have some other numbers also; n is going to denote the number of balls. So we are going to declare it and we are going to read in a value into it and r is going to be the radius of each ball, since we know that the radius does not change, we might as well make it constant.

Then we are going to declare the balls themselves. So each ball is going to be of type or class BALL. This will be defined outside and this is one of the things that has to be done in the assignment. We will declare an array of such balls. Next we will initialize the balls, so that will be a function call and then let us say for simplicity that we have to do this for 100 collisions. So this is the code which is going to do it.

We are going to have a double inside, which is going to be for holding the collision time, then we are going to have an integer, which is going to be an index into the ball array, which is going to be used for holding, which left ball collides with the successive ball. The students will be required to write a function called first collision, in which they actually calculate all pairs of collisions, well actually all adjacent.

So they are only going to consider adjacent pairs of balls. So they are going to consider all adjacent pairs of balls and calculate the collision times between them and the minimum over

those times is going to be returned in the argument C time, which is of course going to be reference argument and also if I and I+1 balls collide, then left collider will contain at the end I. So left collider will also be another reference argument.

Then as per the algorithm given, we will move the balls until C time, so we will simulate the movement for C time and that will just mean that we will have to move the balls on the screen. Then we will set the new velocity and then will repeat.

(Refer Slide Time: 18:18)

Details

The class Ball should hold velocities, as well as the graphical object Circle which will be seen in the animation.

`initializeBalls(b,n,r)` should create balls on screen to have radius r, initialize their positions and velocities randomly.

`firstCollision(b,n,r, cTime, leftCollider)` should determine which pair collides first. If balls i, i+1 are determined as colliding first, i should be returned in leftCollider. The time to collision should be returned in cTime.

`moveBalls(b,n,r)` should move all balls forward along their trajectory for cTime time units.

`setNewVelocity(b,n,leftCollider)` should exchange the velocities of balls leftCollider, leftCollider+1.

So the details should be explained and they are something like what we said earlier, the last ball should hold velocities as well as the graphical object circle, which will be seen in the animation. Initialized balls should create balls on screen to have radius r, so this will contain the statements for constructing balls, so new circle something like that and the positions and initial velocities will be set randomly.

First collision is a function to be written, which is going to determine which pair of balls collide first. If balls I and I+1 are determined as colliding first, I should be written in left collider. The time to collision should be written in C time. Move balls as we said, should just move balls forward along their trajectory for C time, time units and set new velocity should do the exchange of the velocities. So that is about it.

(Refer Slide Time: 19:17)

Remarks

- ▶ This was an actual assignment from one course.
- ▶ The actual problem statement contained more detail, i.e. the final positions were to be reported as result, how to use pseudo random number generators to set positions and velocities, and more specifically what seed to use so that autograding could be used.
- ▶ The leftmost and rightmost balls were required to be held fixed – this ensured that a collision was always possible.
- ▶ Students were also told that for the purpose of testing their code they should use a small number of balls, e.g. 3.
- ▶ One student actually stopped by and commented, “Sir, I liked the way you had broken down the problem for us.”
- ▶ The text [Ran14] often uses this idea: start with a main program that we must write and then fill out whatever is needed.



We have to say a little bit more, so the actual problem statement contains a little bit more detail. For example, you have to say what has to be reported eventually. In our case, we said that the final positions of all balls have to be reported and there was a pseudo random number generated to be used for setting the positions. So we said how to use pseudo random number generator. In fact, we also said what seed should be given.

If we know what seed is going to be given, then we know exactly what the final position is going to be and then we can have an auto grading program. So we can run the program and we can expect exactly the same result for everyone and so that is why we also specified the seed. There was an additional convenience that we use, which is that we said that the left most and the right most balls were required to be held fixed.

This essentially ensured that the balls did not get escaped and 100 collisions were always possible, otherwise what might happen is that the left most balls and the right most balls could escape and eventually all balls could escape from the screen area. We have to help the students and we told in the assignment that look for the purpose of testing just use 3 balls, so may be the 2 balls which are going to hold, fixed and just 1 ball.

So that then if you have to debug you will not be drowned in too much data. So this kind of help you should certainly give, because invariably some students will declare an array of 100 from the

very beginning and then we will just get confused, if there is a mistake. After this assignment, 1 student actually stopped by and commented, Sir, I liked the way you have broken down the problem for us.

This was exactly one of the interesting point, or the main important point of this assignment that if you are writing a large program you should really break it up. Again instead of just telling them to break it up, if you show it, if you show it to them interesting exercise and students are more likely to remember it and they are more likely to be persuaded of its utility. The text book that we have been talking about often uses this idea. They start with a main program and then they flush out whatever else is needed later on.

(Refer Slide Time: 22:07)


Example 2: Working with algebraic expressions

When I learnt programming, it created an impression in my mind that "computers are good with arithmetic".

By providing graphics, we can reassure the students that "computers are good with geometry".

Several years after learning to program, I read the book "Structure and Interpretation of Computer Programs" [AS96] which showed how computers can do symbolic algebra.

- ▶ How to represent formulae such as $x^2 + 2 \sin(x)$.
- ▶ How computers could manipulate the formulae, e.g. print the derivative $2x + 2 \cos(x)$.

 I knew that in principle computers will do anything, but seeing them manipulate algebraic expressions gave me a thrill!

The second example for medium-sized programs comes from working with algebraic expressions. Now when I learn programming in college it somehow created an impression in my mind that computers are good with arithmetic. Of course computers do arithmetic, but now if we work with graphics and we are providing graphics to our students, we can assure the students that is computer is good with geometry.

I mean, we can tell them, that look computers are good with geometry, but that is not as convincing as actually getting them to work with geometry. So for that if you give them graphics, they will themselves know that look computers are good with geometry as well. Now in my case,

several years after learning to program, programming was in languages like FORTRAN and C, I read the book structure and interpretation of computer program.

This book which is very nice by the way and it teaches programming using the language scheme. One of the first things that this book talk about is how computers can do symbolic algebra, what do I mean by this, well they talk about how to represent formulae such as $x^2 + 2\sin x$. And then they also explain things like, how computers could manipulate the formulae. So maybe they would write a program to print the derivate of an algebraic expression.

So if this expression was fit to that program the program would print a derivate $2x + 2\cos x$. Now in principle, I knew that a computer can do anything, but when I actually saw this happening, I thought that this was really amazing, I mean I thought that programs could computers could work with numbers, but somehow I had these feelings that oh, working with algebra is some kind of a superior activity.

Which in a sense it is because we learn about arithmetic in primary school and we learn about algebra in secondary school and that is considered to be a more difficult topic. And here was a computer program also doing algebra, so that give me a thrill, and I think we should try to give this thrill, or we should give this sense of power of computer to our students as well. So what I am going to show you is a program for dealing with algebraic expressions, which you can show to your students, and this is discussed in our book as well.

(Refer Slide Time: 25:07)

A program for dealing with algebraic expressions

Chapter 23, [Ran14]

"Write a program that takes an algebraic expression involving sums and ratios and draws it out on the screen with ratios shown as $\frac{\text{numerator}}{\text{denominator}}$.

Numerators and denominators could be arbitrary expressions themselves
($x/(1 + (1/x))$) should appear as $\frac{x}{1+\frac{1}{x}}$.

Remarks:

- ▶ Problem is vaguely defined – exact appearance is not specified.
- ▶ Encourage the student to discover the specification by taking examples.
- ▶ The specification will be recursive, and the algorithm will immediately follow.
- ▶ **Difficult problem. Interesting use of recursion.**
- ▶ Once student masters this, other problems, e.g. taking derivatives or products are easier.



Taking products of derivatives could be homeworks.

So what is this program going to do, or what is the assignment that you could describe this as. So the assignment could be write a program that takes an algebraic expression involving sums and ratios, and draw it out in the screen with ratios shown by writing the numerator out, putting the bar below it and then putting the denominator under it. So our standard way of writing things, and the important point is that the numerator and denominator could themselves be arbitrary expressions.

So here is an example, so if you have x upon $1+1$ over x , it should appear in this fashion. So look, so there are two levels of division and they are shown by bars of appropriate size. So a few tricky things are here, we have to decide where to position the numerator and the denominator, what size bars to put in between. Now the problem as such, if you give it exactly as given over here, and that is how you should to give it, it is somewhat vaguely defined.

We have not specified exactly how the formulae should appear and there is some artistic freedom that is there, which you really want students to discover. So let students take some example and discover this specification, so maybe they will say, I want to put and so they might say something like, oh, the length of the bar should be as large as the length of the strength, as well as the length of the numerator, as well as the length of the denominator.

In other words, it should be large as the larger of the 2. And then the position also they will have to say, so for example the key question is, how should the bar in $1/x$ be align as compared to the plus operator. So these are some of the things that they need to think about. Now you will see, if you think about it, that the specification itself is going to be recursive, and once you have the specification written down in a recursive manner, then the algorithm will follow in a very natural manner.

I should warn you this is a bit of a difficult problem, so you really should do it in class. It has interesting use of recursion; it is an important problem with that sense as well. Once the student masters this and the student should master it, because there are a few interesting ideas but they are all intuitive. They all have to do with things which you normally do when you write down expressions, exactly where to position, how long to draw the bars, and so on.

So after the student masters this, after the student is able to do the recursion involved in drawing this, then the student will be able to easily do things like given an expression, find out its derivative or may be simply that expression, things like that.

(Refer Slide Time: 28:13)

Algorithm

Simplified specification:

- ▶ All numbers and variables in the formula will be single characters.
- ▶ All operators will be parenthesised, i.e. $(a + b * c)$ will be given as $(a + (b * c))$.

Now it can be solved by novices.

Parsing: A formula is either

- ▶ A letter or a number, or
- ▶ Has the form (formula1 operator formula2).

Recursive structure can be used to develop a reasonably simple recursive algorithm to create an expression tree representing the formula.

Discussed at length in Chapter 23, [Ran14]

Drawing: A recursive algorithm is needed to decide the position of the various parts of the formula. After this another recursive algorithm draws, by referring to the

calculated positions.

So what is the algorithm, the simplified specification is going to be something like this. All numbers and variables in the formulae will be single characters. So we want, if you are going to put a number they will be just be single digit. All operators are going to be parenthesized. So if

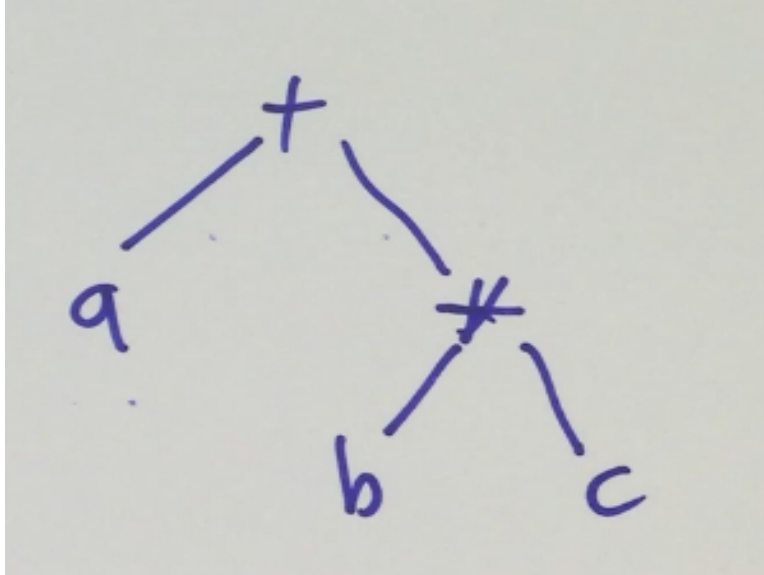
you want to write $a+b$ times c , we will not rely on operator precedence, but we explicitly put down parenthesis.

Once you make this simplifying assumption, then even first year students will be able to write recursive program to solve this. It will be a slightly tricky program, you can help them a little bit, but it will not be hard to understand. If your variables can be longer, if you can have big numbers, then reading in the input itself is going to a difficult problem. So now by simplification we have eliminated some of the unnecessary complexities.

So how do you read in the data, or how do you parse the formula. So the simple observation is a formula is either a letter or a number. So if you see a letter or a number you know it is a formula. If you see an open parenthesis, then you know after that there should be a formula, followed by an operator, followed by another formula, followed by a parenthesis. So formula 1 and formula 2 will potentially be read by recursion.

So this will give raise to a reasonably simple recursive algorithm, actually the code is extremely simple, the thinking process might take some while to get used to. And the recursive structure itself can be used to develop, recursive algorithm, not just to read the problem, but while reading convert it into an expression tree to represent the formula. So for example, this formula that I have written down.

(Refer Slide Time: 30:26)



It could be represented as plus which is the root a and product of b and c. So you will be able to write a program, which will read that formula and generate this expression tree. Now drawing is not easy, but again there is a recursive algorithm needed, this has been explained and it will calculate the positions of the various parts where we are supposed to draw the various parts and this calculation will have to be recursive and after this another recursive algorithm will draw by referring to the calculated positions.

So as I said, this is a little tricky, but once you do it, your students will be able to do many other things, because the other things are likely to be simpler than this actually. So it is like this, if you want students to drive a car at 50 kilometres per hour, and you take them on a ride and you show them how to drive safely at 100 kilometres per hour, then they will be quite comfortable in going anywhere in 50 kilometres per hour.

So something like this, sort of give them, sort of a heavy dose and then the simpler more normal routine things they will be able to do by themselves.

(Refer Slide Time: 31:59)

Remarks

- ▶ Representation of a formula: flag to say whether it is a variable name or a number, or consists of an operator and (pointers to) subformulae.
- ▶ If OOP is taught, in C++ formulae may be represented using a class hierarchy.
 - ▶ A parent class Formula, possibly abstract.
 - ▶ Subclasses NumericFormula, variableFormula, sumFormula etc. inheriting from Formula.

Discussed in Chapter 26, [Ran14] with a deeper hierarchy.

Useful to show the same program written with and without inheritance.

- ▶ Once students understand how to represent and manipulate formulae, they can perform operations such as taking sums, products, derivatives, and performing simplification.

Opens up a new vista for the student.



- ▶ Bright students will also perform general parsing and implement operator precedence.

So some remarks, so the representation of a formula is itself an interesting question. So we will use abstract of course or a class and we will have to use a flag to say whether the abstract is going to hold the name of a variable or a number or it will hold pointers to that formulae and the operator. So actually we will have to allocate space in the abstract to hold all these things. Now if object oriented program is start.

We are going to talk about object oriented programming in a few minutes and if you are using C++, then you can represent the formula using a class hierarchy, and if you do that, then the object that you create for each formula say if you know that a formula is just a variable name, then it will not need to contain fields to hold pointers to sub-formulae, it will just have enough space to hold the variable which that formula is or the number which that the formula is.

So potentially the way this will be done is that we will have a parent class called formula which could possibly be an abstract class and then we will have sub classes, one for the numeric formula, one for variable formula may be one for some formula or product formula, division formula, and this will inherit from formula, if you know object oriented programming, this is a very natural way of doing things.

Now you should actually consider writing the same program in both ways, because then the students will be able to see the difference and they will be able to see the utility of object oriented

programming. So one of the questions that we have talked about with great emphasis is that we have to tell students why something is useful.

So if you want to tell students object oriented programming is useful, then perhaps it is a good idea to show them the same program written, without object oriented programming, with object oriented programming and then explain to them the benefits. Now once our students understand how to represent and manipulate formulae, they can perform operations, many other operations say may be taking sums of formulae, products, derivatives, performing simplifications and they can write a whole program which manipulates formulae.

You might have seen some commercial programs which do this, for example there is a program called mathematica and there are several others of course, which does things like this. Of course your students will not be immediately able to write programs like professional programs, but at least they will know and some of them will be inspired to do big things. So essentially by teaching this you are opening up a big new vista, a big new area for you students.

Now when we discussed all this we said that we are going to pose a simplified problem, which is that we are going to have variables names be exactly one character, we are going to have number with exactly one digit. But you can give as homework assignments the problem of extending the program for general parsing, and you will see that once you go half the distance, your students will want to go the rest of the distance, and they will implement general parsing as well as operator precedence and things like that.

So again if you tell them that the idea is first think about how you do things manually. They will be able to do it although, if you look at compiler courses, they have a very special way of implementing operator precedence. Your first year students, your introductory programming students may not be able to discover that way and they may not be able to discover the most efficient way, but they will discover one full proof way of doing this simply by thinking about how they themselves implement and understand operator precedence.

So for example, they will say that look, I know that multiply is higher than plus, so first I will go through my entire program and see which multiplies I can simplify, so things like that they will be able to do. So I am going to give a few additional examples of medium size programs, so you can do interpreters and editors, so may be students implement circuit or drawing editors.

(Refer Slide Time: 36:51)

More examples of medium sized programs

- ▶ Interpreters/editors.
 - ▶ Examples: Circuit/drawing editor with graphical input output
 - ▶ "Read-eval-print" Loop: read commands, modify state
- ▶ Simulators
 - ▶ Simulation of a physical system
 - ▶ CPU simulator/machine language interpreter
- ▶ Information/transaction processing systems
 - ▶ Library management
 - ▶ Student grade management

"Model view controller"

Ch 19, Ch 28 [Ran14]

Build on Ch 2, [Ran14]

Typically, there is a read-eval-print loop inside this, so the user will type some commands, this will be read by the program and this will modify state or may be modify what is on the screen. This is also sort of like the model view controller pattern, you can have simulators, so simulation of a physical system, so a number of examples are given, you can also have CPU simulator, or a machine language interpreter.

So for this you can build on the chapter 2 of the book and you can have information or transaction processing systems of course, so may be library management, may be student grade management. So we will stop the sub-lecture and then continue in the next sub-lecture. Thank you.