

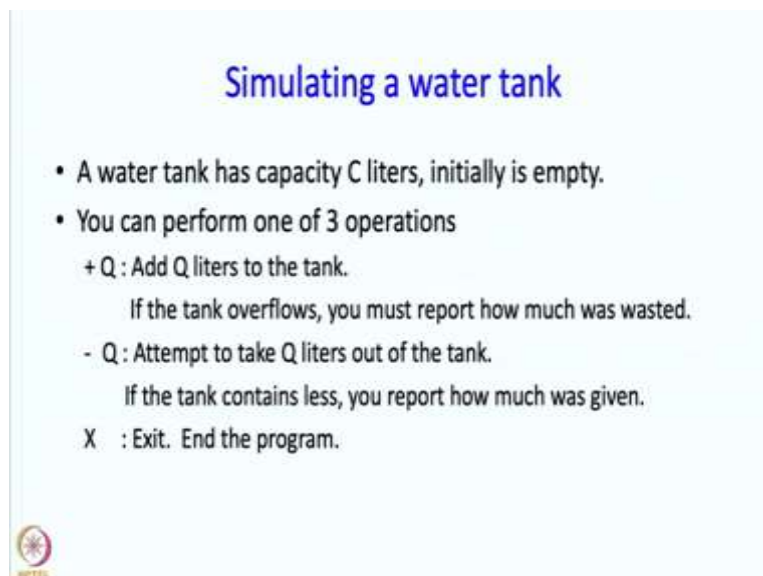
An Introduction to Programming through C++
Professor Abhiram G. Ranade
Department of Computer Science and Engineering
Indian Institute of Technology Bombay
Lecture No. 09: Part - 4
Loops in various applications
Simulating a water tank

(Refer Slide Time: 00:24)



Welcome back in the previous segment we discussed a very simple real life problem, and we wrote a program for it, this was the bargaining problem. Now we are going to consider a slightly more complicated problem which is also resembling a real life problem.

(Refer Slide Time: 0:41)



So, this is about simulating a water tank, so we have a water tank which has capacity some c , c liters and initially it is empty, and you can perform one of 3 operations with respect to this

tank. So, you can add water to it, and let us say this operation will be performed and as far as the computer is concerned to add water into the tank, you will type plus followed by the amount you want to add in litres.

Now, if you add water to a tank it may overflow, so you may be adding too much water. So, in which case, water will be wasted, so you are expected to report, your program is expected to report, how much water is wasted, or you may attempt to take q litres out of the tank. So, somebody say walks to the tank, and says give me 15 litres, now if the tank has 15 litres then you will give 15 litres, but if the tank does not have 15 litres, you will give whatever is there, but you will say how much you actually have given.

And these commands will keep on coming at you, and have to respond to them properly, and note that means that you need to know, at each step how much water is there in the tank. So, this is an example of a system in which there is some state, the state is how much water is present, and what the external world is doing to the tank? Well it is sending some commands, as were result of which the system reacts and it produces some output and in additional its state also changes. So, it may give you some water, it may take some water from you, and at the same time its state will change, and it will printout some messages. So, these are the things that we want our program to do.

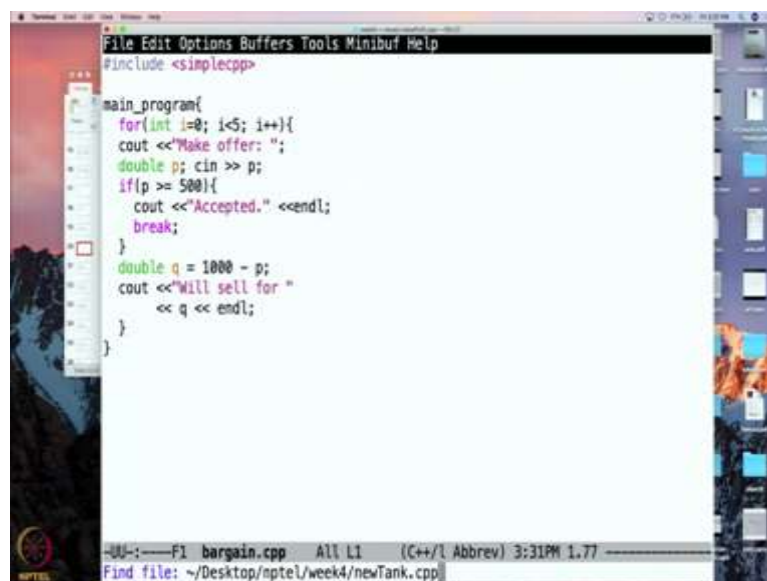
So, this kind of thing happens quite frequently you may, you are trying to keep track of a real life system, and you are trying to see, how it responds as external communication happens to it? So instead of a water tank for example, this might be a bank account, and you may be withdrawing money from it, you may be depositing money from it, it is all here, very similar. And of course, there is an additional command possible x so, if you type x that is a signal to the program that look, stop. Close everything, just exit the program.

(Refer Slide Time: 03:42)



Alright, so now what we are going to do is? We are going to write the code for this, and this time I am going to write it along with you. So, let us see, this going to bit of a long code, but I will write it with you.

(Refer Slide Time: 03:53)



```
File Edit Options Buffers Tools Minibuf Help
#include <simplecpp>

main_program{
    for(int i=0; i<5; i++){
        cout <<"Make offer: ";
        double p; cin >> p;
        if(p >= 500){
            cout <<"Accepted." <<endl;
            break;
        }
        double q = 1000 - p;
        cout <<"Will sell for "
            << q << endl;
    }
}
```

Find file: ~/Desktop/nptel/week4/newTank.cpp

A screenshot of a C++ code editor window titled 'newTank.cpp'. The code defines a 'main_program' with a 'while(true)' loop. Inside the loop, it reads a 'command' character. If 'x', it breaks. If '+', it reads a quantity 'qty' and adds it to 'current'. If 'current' exceeds 'capacity', it prints 'Wasted: ' followed by 'current - capacity' and resets 'current' to 'capacity'. If '-', it reads 'qty' and if 'qty' is greater than 'current', it prints 'Giving: ' followed by 'current' and resets 'current' to 0. The editor's status bar shows 'newTank.cpp All L24 (C++/1 Abbrev) 3:40PM 1.63'.

```
File Edit Options Buffers Tools C++ Help
#include <simplecpp>

main_program{
    double capacity = 30, current = 0;

    while(true){
        char command;
        cin >> command;
        if(command == 'x') break;
        else if(command == '+'){
            double qty;
            cin >> qty;
            current = current + qty;
            if(current > capacity){
                cout <<"Wasted: "<<current - capacity<<endl;
                current = capacity;
            }
        }
        else if(command == '-'){
            double qty;
            cin >> qty;
            if(qty > current){
                cout <<"Giving: "<<current<<endl;
                current = 0;
            }
        }
    }
}

-UUJ:~--F1 newTank.cpp All L24 (C++/1 Abbrev) 3:40PM 1.63
```

So, let me try it out, so let me open a new file, say newtank.cpp. So let me put down `#include<simplecpp>`, (ohh what am I doing, so 1 m) then let start the main program. So, how we go about this, what do we need to, what variables do we need to have? Well, we need to clearly keep track of how much water is there, in our tank. So, we need to know what is the current content. So we will need the variable that, and we also would like to have, we also would like to know, what is the capacity, so we will have variables for both of these.

So, let us just for fun, make the capacity be 30 litres and then, let us have a variable say called current, and let say initially it is empty, so this variables current tell you how much water is there in the tank at the current time. Alright, now, so these are the sort of main variables we need, this is not a variable capacity is not a variable, but instead of putting in 30 everywhere in our program, it is better to give it a name. So, could have even made this a constant, but never mind, it is so important right now.

So, what happens next? So now, we are going to simulate the execution of the system. So this has to be done as a while loop, we just keeps running, so while, and since it keeps running we are not yet ready to decide whether it to terminate it, we will give the condition as being true. So, what is the first thing we do in each iteration? Well, in each iteration we have to ask the user to type something so, we have to expect the command from the user. So, that command is going to be a single letter command followed by possibly a number, so we want to have some variable in which to receive that command. And so, it is customary to have say char command, and then we will do `cin into command`.

So, at this point we have got the first letter that the user has typed into our variable called command. So, if that letter is x then we have to exit. So let us just put that down, so if

command==x and let us say it is just lower case x, then we will break, so break and that will be the end of the program as well, there will be nothing following this loop.

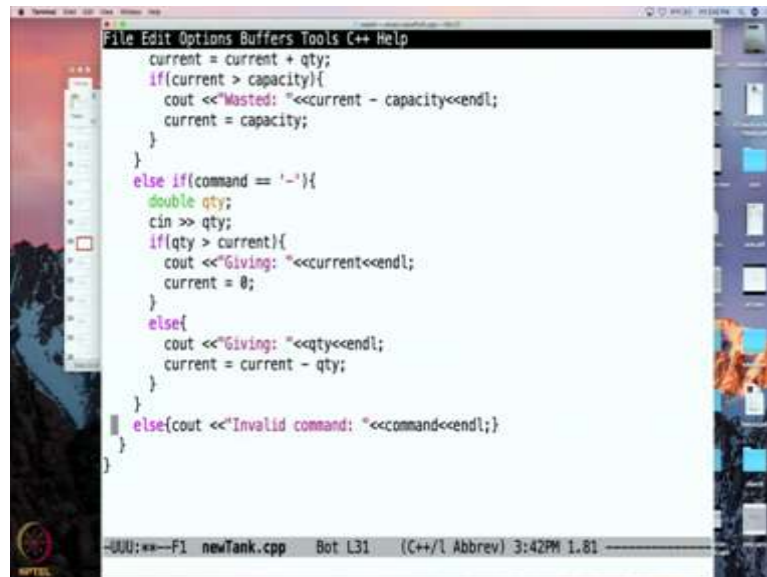
Otherwise, we should check whether the user wants to add water, or the user wants to take out water. So if command is equal to say plus, then what should you do, just we want to put over here, so if the user typed a plus, then we know that the user is going to add water, and so we should expect the user to type in a number. So, for that we will have int some quantity and we will say cin into quantity. Now here, I am assuming that quantity is going to be an integer, sorry lets make it double, we said this is supposed to be double, so our capacity is double, so might as we will make this double. But we want this quantity to be a positive number, so if somebody types add, I want add negative minus 30, so this is probably something wrong, so we should probably be checking. But lets say that such mistakes are not made, and after a plus the user will be sensible enough to type in a positive number. So then this is fine.

So, what should we do at this point? So the user wants to add qty into current, so there is the possibility of an overflow, and we are expected to report, how much overflow is there. So, here is what I am going to do, I am just going to first add it to current, so I am going to say, $current=current+qty$. Now I will check, if current equals, sorry if current is, greater than capacity then what should I do? Now I have to printout, how much water is wasted, so how much is wasted? So, the amount of wastage is, so how much is wasted? Well, everything above the capacity is wasted. So, this quantity the wasted quantity is current minus capacity. Is that all? Well no, our current is above capacity and we just removed something out of it, so we should leave it at the actual current value, so when we port it went above capacity, but we know that it cannot go above capacity, so whatever goes above capacity is wasted. So, we should leave current equals capacity. If current is less than or equal to capacity, we have to do nothing. And therefore, that is enough.

So, this is the processing we need to do, if our command was a plus, else if command is minus, what do we need to do? Well, again we need to do something similar, we are expecting the user have to type in a number, a quantity. So, let us do that, so double qty and cin into qty. So the user typed in something, and may be the user has demanded something more than what we have available. So I guess at this point, we should check. So, we should say if qty is bigger than current, then what happens? So, if the quantity demanded is bigger than the current quantity, then how much are we going to give? So, we are going to give, giving qty. And we should change the current values, so how much value remains? Well, we

are giving, sorry we are not giving qty, we are giving current because qty was bigger, so how much remains? So then current is equal to 0.

(Refer Slide Time: 13:47)



```
File Edit Options Buffers Tools C++ Help
current = current + qty;
if(current > capacity){
    cout <<"Wasted: " <<current - capacity<<endl;
    current = capacity;
}
}
else if(command == '-'){
    double qty;
    cin >> qty;
    if(qty > current){
        cout <<"Giving: " <<current<<endl;
        current = 0;
    }
    else{
        cout <<"Giving: " <<qty<<endl;
        current = current - qty;
    }
}
else{cout <<"Invalid command: " <<command<<endl;}
}
}
```

--UJL:***--F1 newTank.cpp Bot L31 (C++/l Abbrev) 3:42PM 1.81

So this is what happen if qty is bigger than current, what if it is less than or equal to current? In which case, how much are you going to give? I should have not typed semicolon over here, but a colon, how much are we giving? Well, in this case, we are giving whatever is demanded. And what is the new value? So this is equal to current minus what we give. So, that is it. So, that is the end of this minus command processing.

Now here, may be the user types in some other character, so maybe we really should put an else, cout<<"invalid command"; And that is the end of the loop. So, this is going to get repeated and this if you exit out of it, that is going to end the program.


```
Terminal [1] emacs naivePyth.cpp
~/Desktop/nptel/week4 : s++ bargain.cpp
~/Desktop/nptel/week4 : ./a.out
Make offer: 700
Accepted.
~/Desktop/nptel/week4 : ./a.out
Make offer: 50
Will sell for 950
Make offer: 20
Will sell for 900
Make offer: 200
Will sell for 800
Make offer: 400
Will sell for 600
Make offer: 600
Accepted.
~/Desktop/nptel/week4 : %
emacs naivePyth.cpp

[1]+ Stopped                  emacs naivePyth.cpp
~/Desktop/nptel/week4 : s++ newTank.cpp
~/Desktop/nptel/week4 : ./a.out
Currently we have: 0
+ 50
Wasted: 20
Currently we have: 30
```

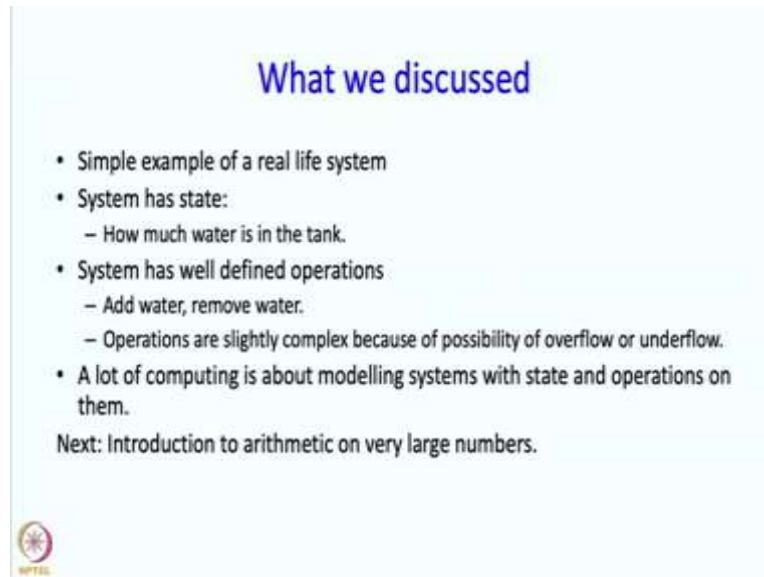
```
Make offer: 400
Will sell for 600
Make offer: 600
Accepted.
~/Desktop/nptel/week4 : %
emacs naivePyth.cpp

[1]+ Stopped                  emacs naivePyth.cpp
~/Desktop/nptel/week4 : s++ newTank.cpp
~/Desktop/nptel/week4 : ./a.out
Currently we have: 0
+ 50
Wasted: 20
Currently we have: 30
- 15
Giving: 15
Currently we have: 15
-20
Giving: 15
Currently we have: 0
-10
Giving: 0
Currently we have: 0
+25
Currently we have: 25
x
~/Desktop/nptel/week4 : █
```

So, let me try compiling it, it seems to have compiled, no errors may be I did not make any errors while typing it out. So it is telling me currently we have 0, so let us say, let me add something. So let me add may be 50, so it says you wasted 20, currently you have 30. Is that right? Well, it is because the capacity of that tank is 30. So maybe let us take away something, so let say minus 15, so it is giving 15 and what remains is 15. So let us try minus 20, so you asked for 20, but it is only giving you 15, is that reasonable? Well yes, it only has 15. And after giving you 15, it only has 0. Which is good, which is as you might expect. So, let us try to take a little bit more, so may be minus 10. So it is not giving you anything, because it does not have anything. And it continuous to have nothing. So maybe we will add say, plus 35. Let us say let us add plus 25, so it says currently we have 25, but notice that it did not printout a message saying, something was wasted, because nothing was wasted.

So, this seems like, a reasonable, a correct program, so we can get out of it. Let see we will be get out of it, yes, so that is it, so that is this program, so let me get back to the presentation now.


(Refer Slide Time: 18:28)



What we discussed

- Simple example of a real life system
- System has state:
 - How much water is in the tank.
- System has well defined operations
 - Add water, remove water.
 - Operations are slightly complex because of possibility of overflow or underflow.
- A lot of computing is about modelling systems with state and operations on them.

Next: Introduction to arithmetic on very large numbers.



So, we did this, so, what did we discussed in this segment? So we discussed a simple example of a real life system, this system does have state. How much water is in the tank, and the system has well defined operations, add water or remove water, and the operations are slightly complex because of the possibility of overflow and underflow. So, I want to point out that this may seem like a trivial system, and in a sense it is, but it is an example of a system which has a state and which has operations allowed. So you will see systems which are just more complicated versions of this, and there is a basic loop, when dealing with such a system, that you wait for the environment to say something, may be make a request, then the system responds to it, and prints out the message and it changes its state. Alright, so that is what we wanted to see in this topic, in the next segment we will talk about different topic that of arithmetic on very large numbers. So, we will take a break.