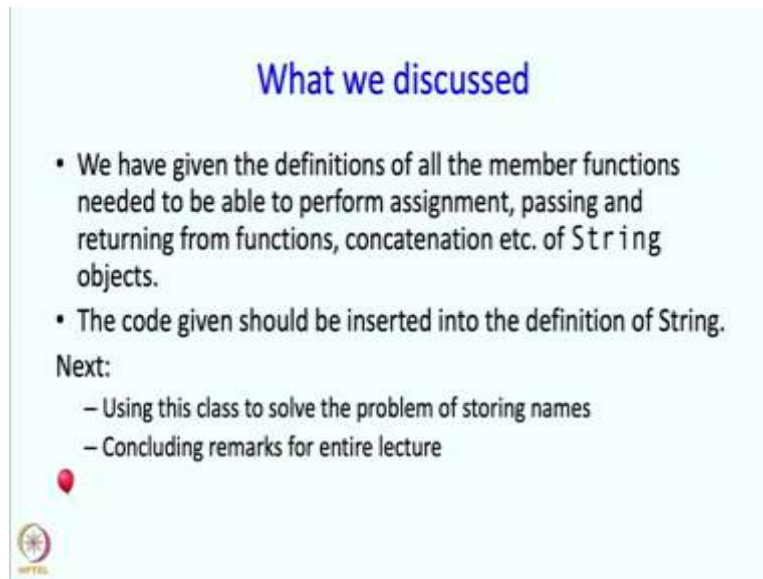


An Introduction to Programming Through C++
Professor Abhiram G. Ranade
Department of Computer Science and Engineering
Indian Institute of Technology Bombay
Lecture 22 Part-7
Representing variable length entities
Using the implemented class and conclusion

Welcome back. In the last segment we gave the definitions the complete definition of our string class.

(Refer Slide Time: 00:25)




What we discussed

- We have given the definitions of all the member functions needed to be able to perform assignment, passing and returning from functions, concatenation etc. of `String` objects.
- The code given should be inserted into the definition of `String`.

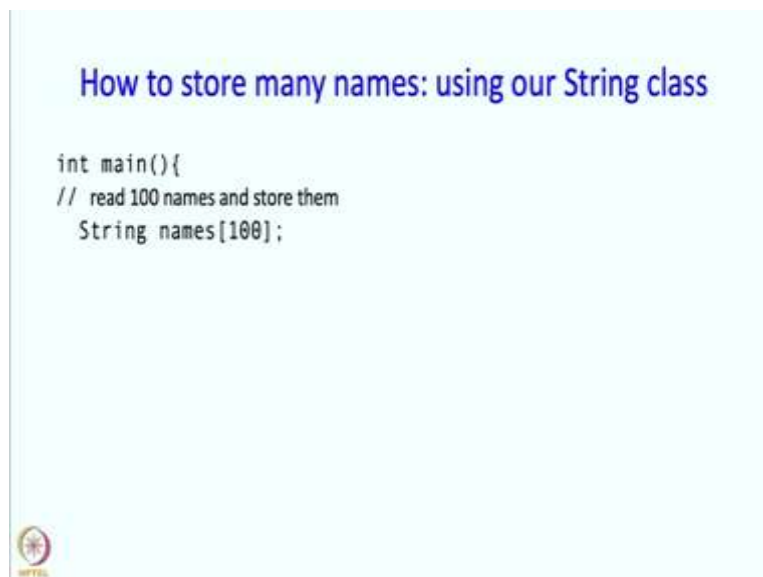
Next:

- Using this class to solve the problem of storing names
- Concluding remarks for entire lecture




Now, we are going to use that first to solve the problem with which we started this lecture.

(Refer Slide Time: 00:34)



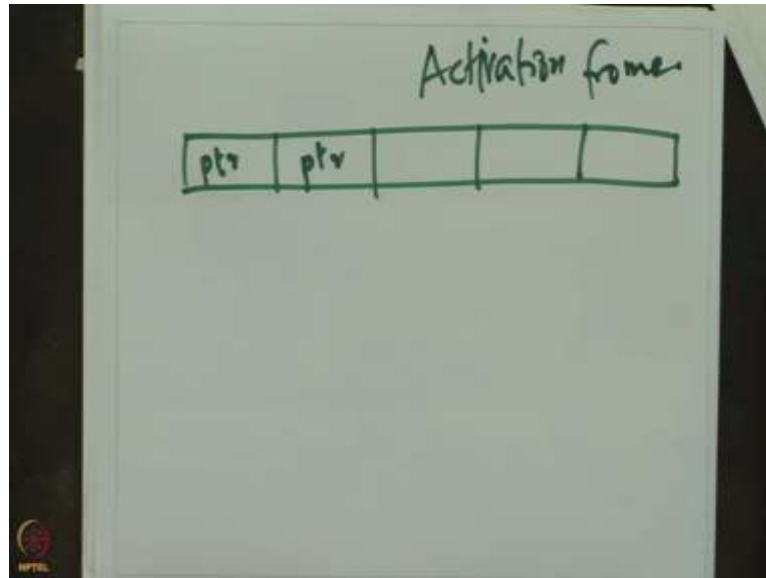
How to store many names: using our `String` class

```
int main(){  
  // read 100 names and store them  
  String names[100];  
}
```



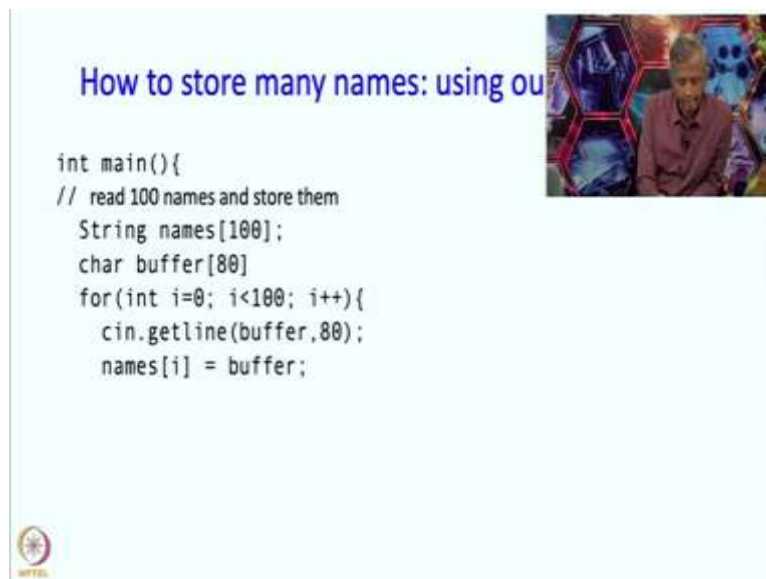
So, the problem was how to store many names of varying sizes. So, here is the program, so we are going to read 100 names and store them. So, for that we are going to allocate an array of strings. So, it should be instructive to just see what this actually does. So, in the main programme this is going to create an array of string objects.

(Refer Slide Time: 01:03)



But let me remind you that a string object is really very simple, it just contains a single PTR, so this that PTR. So, we have 100 PTRs really because each object just contains one PTR. So, 100 words will be allocated in the activation frame. And this PTRs will be null by the way, at the beginning.

(Refer Slide Time: 01:45)

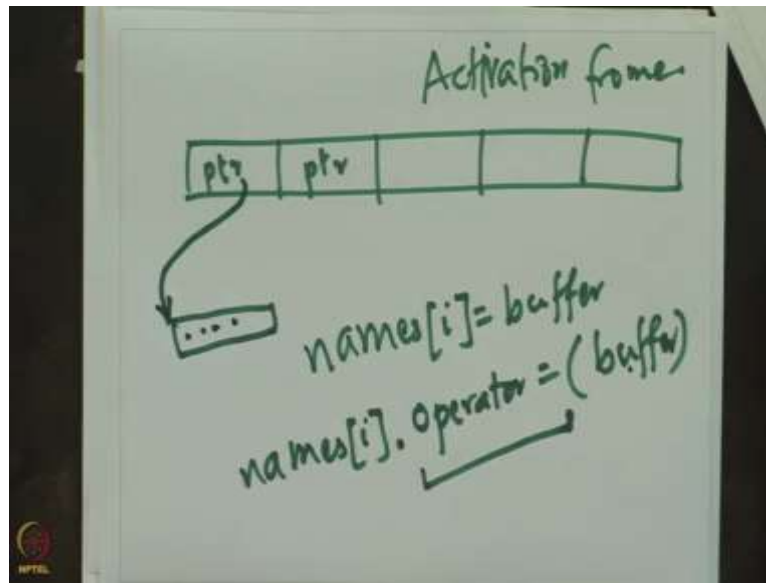


Then to do the reading of the names we are going to use this buffer of 80 characters. This is just meant to be a large enough number and now we are going to read each of the 100 names. So, in this

loop iteration we are going to that. So, this time let us say we use the safe alternative but this will also allow us to get in the spaces.

So, this will cause the name to be read from the keyboard new line terminated into our char array buffer and of course inside the buffer it will be stored with a null termination and then we are just going to assign `names[i]=buffer` that is it.

(Refer Slide Time: 02:34)



So, if you remember how is this going to work, `names[i]=buffer`. So, this is going to be viewed by C++ as `names[i].operator=(buffer)`. And if you remember we defined this operator equal to a minute ago and operator equal to which takes a char array as argument. We defined that a minute ago and so what that will do is it will allocate some space on the heap and it will start pointing this pointer to this and it will copy whatever was in the buffer into the space.

(Refer Slide Time: 03:33)

How to store many names: using our String class

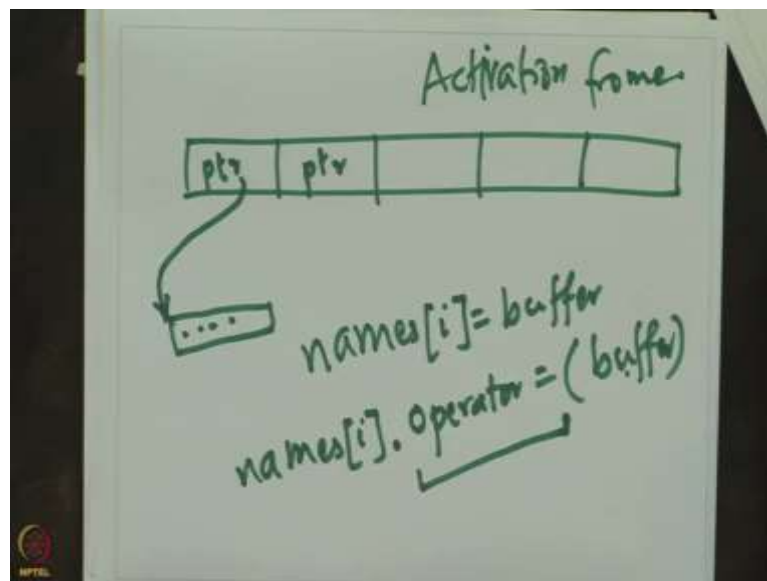
```
int main(){
// read 100 names and store them
String names[100];
char buffer[80]
for(int i=0; i<100; i++){
cin.getline(buffer,80);
names[i] = buffer;
}
// now use names[]
}
```

- The number of bytes used for `names[i]` will be equal to 1 + the number of characters in the name.

So, in this code we are not mentioning things like how many characters did we read into buffer but all that will happen as a result of this assignment, so that is it, that is the code. Now, you can use this names array however you want, you can do whatever you want with the names inside that array.

So, again just want to point out that we have solved the problem that we started of this lecture with the number of bytes used for names[i] will be equal to 1 plus the number of characters and incidentally this was exactly the solution we got even in the solution that we got very early on in the lecture, we had the same situation in the activation frame, we had an array of pointers.

(Refer Slide Time: 04:22)




Even here we effectively have an array of pointers although we called it an array of string objects, the string objects themselves are containing only one member which is a pointer and that pointer points to something on the heap on the names which are stored on the heap. But the beauty is that over here in this solution, we are not talking about heap at all. We are not talking about how many characters we are transferring, how much we are allocating and anything like that.

(Refer Slide Time: 04:45)

How to store many names: using our String class

```
int main(){
// read 100 names and store them
String names[100];
char buffer[80]
for(int i=0; i<100; i++){
    cin.getline(buffer,80);
    names[i] = buffer;
}
// now use names[]
}
```

- The number of bytes used for names[i] will be equal to 1 + the number of characters in the name.
- But you do not have to mention these details, they are implemented automatically!
- In fact, if we use our class String, we do not need to mention memory allocation, it happens automatically in the member functions.



So, all this is implemented automatically. And we are not mentioning memory allocation. And we do not have to worry about memory leaks, we do not have to worry about dangling pointers, our implementation of string class takes care of all of these things.

(Refer Slide Time: 05:11)

Demo

- numWords.cpp



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

int length(const char* S){
    int L=0;
    while(*S !=0){S++; L++;}
    return L;
}

void scopy(char* d, const char* s, int L=0){
    d = d+L;
    while(*s != 0){
        *d = *s; d++; s++;
    }
    *d = 0;
}

class String{
    char* ptr;
public:
    String(){ // constructor
        ptr = NULL; // initially empty string
    }
    void print(){ // print function
        if(ptr != NULL)
            cout << ptr;
        else
            cout << "NULL";
    }
    // other member functions...
};

// P1 nmWords.cpp Top L1 (C++/L Abbrev) 4:13PM 1.53
Mark set
```

So, I am now going to give a quick demo which uses our string class. In this program we did not use the concatenation feature but in this demo we will be doing so, so let us take a look at that. So, this is words, so here I have copied the length and scopy functions and then over here our string class starts. The string class is here with the member functions.

(Refer Slide Time: 05:46)

```
File Edit Options Buffers Tools C++ Help
}
// other member functions...

String& operator=(const char* rhs){
    // release the memory that ptr already points to.
    delete[] ptr;

    // make a copy of rhs on the heap
    // allocate length(rhs) + 1 byte to store '\0'
    // Assume length function (Section 15.1.4)
    ptr = new char[length(rhs)+1];

    // actually copy. Function scopy from 15.1.4
    scopy(ptr, rhs);
    return *this;
}

String& operator=(const String &rhs){
    if(this == &rhs) return *this;
    // If it is a self assignment, we do nothing.

    delete ptr; // @release the memory that ptr already points to.

    // Make a copy of rhs.ptr on the heap
    // Allocate length(rhs.ptr) + 1 byte to store '\0'
    ptr = new char[length(rhs.ptr)+1]; // length: Section 15.1.4
}

// P1 nmWords.cpp 18% L27 (C++/L Abbrev) 4:14PM 1.82
```

```
File Edit Options Buffers Tools C++ Help
ptr = new char[lengthrhs.ptr+1]; // length: Section 15.1.4

scopy(ptr, rhs.ptr); // Actually copy. Function scopy from 15.1.4
return *this;
}

char& operator[](int i){
    return ptr[i];
}

String operator+(const String &rhs){
    String res; // variable for constructing the result
    res.ptr = new char[length(ptr) + length(rhs.ptr) + 1];
    //allocate needed space

    scopy(res.ptr, ptr);
    // Copy the string in the receiver into the result.

    // Copy the string in rhs into the receiver, but starting at
    // length(ptr), i.e. after the string copied earlier.
    scopy(res.ptr, rhs.ptr, length(ptr)); // scopy is given in book.

    return res;
}

~String(){
    delete ptr;
}

F1 numWords.cpp 45% L53 (C++/L Abbrev) 4:14PM 1.02
```

So, all the operators, everything is there and this is the destructor which we also wanted.

(Refer Slide Time: 05:59)

```
File Edit Options Buffers Tools C++ Help
}

String(const String &rhs){
    // rhs = argument, receiver = parameter in F
    ptr = new char[lengthrhs.ptr+1];
    scopy(ptr, rhs.ptr);
}

};

int main(){
    String units[10], tens[10];
    units[0]="";
    units[1]="one";
    units[2]="two";
    units[3]="three";
    units[4]="four";
    units[5]="five";
    units[6]="six";
    units[7]="seven";
    units[8]="eight";
    units[9]="nine";

    tens[0]="";
    tens[1]="ten";
    tens[2]="twenty";

F1 numWords.cpp 74% L102 (C++/L Abbrev) 4:14PM 1.02
```



```
int main(){
    String units[10],tens[10];
    units[0]="";
    units[1]="one";
    units[2]="two";
    units[3]="three";
    units[4]="four";
    units[5]="five";
    units[6]="six";
    units[7]="seven";
    units[8]="eight";
    units[9]="nine";

    tens[0]="";
    tens[1]="ten";
    tens[2]="twenty";
    tens[3]="thirty";
    tens[4]="forty";
    tens[5]="fifty";
    tens[6]="sixty";
    tens[7]="seventy";
    tens[8]="eighty";
    tens[9]="ninety";

    String dash; dash = "-";
    int z; cin >> z;
    (tens[z/10]+dash+units[z%10]).print();
    cout << endl;
}
//=====FI namWords.cpp 88% L117 (C++/I. Abbrev) 4:15PM 1.09
Auto-saving...done
```

So, what is our problem? So, the problem that we are going to look at is sort of a fun problem, nothing useful by any means but just a fun problem. So, what is the problem? So, I will tell you what are we going to do. So, this program is going to read in a number from the user first and then it is going to print out something, so that is about what it is going to do.

So what is it going to print out? So it is going to read in a number which it expects to be a two digit number. Actually it expects the number to be between 20 and 99, so if you type a number between 20 and 99 say you type 37 then this program will print out 37. So, it just writes the number in words and it does this only for a two digit number but it illustrates the kinds of things that we have been doing over here.

So, what happens here? So, first we have an array of strings by the way once we define strings then the C++ machinery will allow us to define arrays of strings, that is no problem. So, we will have an array of strings, one array called units and another array called tens. So, in units we are going to put 1,2,3,4,5,6,7,8,9. We could put a 0 over here as well but we really do not want to because you will see why we are not putting a 0 and in tens again we are not putting a 0 but we are putting 10, 20 all the way till 90.

So, what is this doing over here. So, you give me a number then I am going to extract the tens digit from it, so that is simply z divided by 10 and then I am going to use that to index into the tens array. So, suppose the number I type was 37 then the tens digit or 37 divided by 10 is going to be 3. So, I am going to index into this and now that will get me my thirty. So, remember that I am supposed to print 37, so I have now been able to get thirty from this. Then I want a dash, so dash is another string called dash which I have assign to this character dash.

So, I am going to add it or I am going to concatenate these two strings together but I also want the unit's place, so to get the unit's place, I just have to do $z \text{ mod } 10$ so for 37 this would be 7. So, units of 7 would be this, so I would get 7. So, from this I would get thirty, from this I would get dash and from this would get 7. So, I would get thirty dash 7 concatenated which I am going to print and since print is not putting an end line, I am going to put an end line myself. So, that is all varies to the program.

It is kind of a trivial program but it does illustrate some ideas one of the ideas is that you can use an array of characters to do some interesting things because you can index into that array but most importantly I am doing this because I want to show you that here we are able to concatenate the objects which we are storing in our string class.

(Refer Slide Time: 09:45)

```

~/Desktop/nptel/week10 : %
emacs -nw vectest.cpp

[1]+  Stopped                  emacs -nw vectest.cpp
~/Desktop/nptel/week10 : s++ numWords.cpp
+ g++ numWords.cpp -Wall /Users/abhiram/simplecpp/lib/libsprite.a -I/Users/abhiram/simplecpp/include -I/opt/X11/include -L/opt/X11/lib -lx11 -std=c++17
~/Desktop/nptel/week10 : ./a.out
9
nine
~/Desktop/nptel/week10 : ./a.out
18
18
~/Desktop/nptel/week10 : ./a.out
46
forty-six
~/Desktop/nptel/week10 : %
emacs -nw vectest.cpp

[1]+  Stopped                  emacs -nw vectest.cpp
~/Desktop/nptel/week10 : s++ numWords.cpp
+ g++ numWords.cpp -Wall /Users/abhiram/simplecpp/lib/libsprite.a -I/Users/abhiram/simplecpp/include -I/opt/X11/include -L/opt/X11/lib -lx11 -std=c++17
~/Desktop/nptel/week10 : ./a.out
87
eighty-seven
~/Desktop/nptel/week10 : %
emacs -nw vectest.cpp

[1]+  Stopped                  emacs -nw vectest.cpp
~/Desktop/nptel/week10 :

```

```

~/Desktop/nptel/week10 : ./a.out
37
thirty-seven
~/Desktop/nptel/week10 : ./a.out
85
eighty-five
~/Desktop/nptel/week10 : ./a.out
34
34
~/Desktop/nptel/week10 :

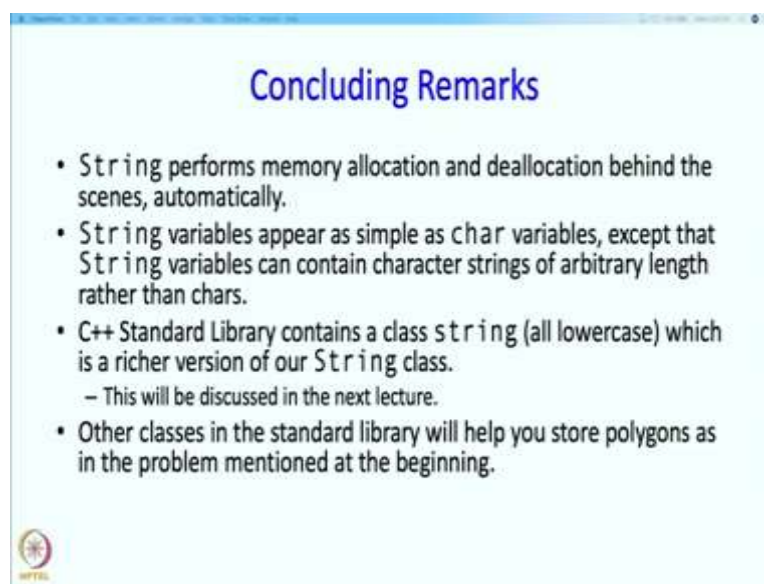
```

So, let us compile this and run it and let us run it. So, let us see we write 37, so we do get 37. Let us do one more, say 85 and we will get 85. But let us do something improper.

So, let us say we type 5. So, let us see what we got, so we got the 5 alright but that 0 in tens place did not come as it should not have come but we did get that dash because that dash is there, so maybe we should write we should modify our code so that dash also does not appear and of course you really should make some special cases, you should put in some if statements to check whether the numbers are 11,12,13,14 all the all the teens basically and for all the teens and ten itself and 11,12 there should be a special case and those cases should also be handled. .

So, anyway, so that is what this program does and as you can see it uses the capital string class that we just created

(Refer Slide Time: 11:09)



And we are really at the end of the lecture, so I just want to make a few concluding remarks. So, our class string performs memory allocation and deallocation behind the scenes and it happens automatically in the sense that the user is not really aware of what is going on. Now, string variables appear as simple as char variables.

So, when you use char variables you do not really need to worry about where are these stored or you could think of them as being stored in the activation frame and if you want you can think of the string variables also has been stored in the activation frame That will not really cause any major problems. The only difference is that the string variables can contain character strings of arbitrary length rather than chars, individual chars.

So, in the next lecture we are going to see a class from the standard library called string where s is not capitalized and we will see that it is a richer version of our string class but the point of discussing this capital string was to see how such classes can be implemented.

So, the basic idea is that these classes that we are going to see that appear in the standard library and some of which we are going to see soon use memory allocation and deallocation from the heap and as an example we have studied this, we have designed and studied this String class and there are other classes in the standard library which we will discuss and one of them will directly enable you to store polygons, variable sized polygons as was mentioned at the beginning of the lecture. So, that concludes this lecture, thank you.