7COM1025
Programming for Software Engineers
Lecture 5

Dr. Renato C. de Amorim

STRINGS (NULL TERMINATED)

These are arrays of characters

```
char str[10];
```

The compiler automatically adds a null terminator (‘\0’). You should declare your strings taking this into account.

Note that cin stops in at the first whitespace! You can use gets() from <cstdio>

```
#include <iostream>
#include <cstdio>
using namespace std;
int main()
{
	char str1[80], str2[80];
	cout << "enter two strings (with a whitespace in each)"<<endl;
	gets(str1);
	cin >> str2;
	cout<<"String 1: "<<str1<<endl;
	cout<<"String 2: "<<str2<<endl;
	return 0;
}
```

STRINGS (NULL TERMINATED)

You can use the null-terminator in loops

```
#include<iostream>
#include<cstdio>
using namespace std;
int main()
{
    char str[20];
    int i;
    gets(str);
    for (i=0; str[i]; i++)
        cout<<str[i];
    cout<<" The Null terminator is in: "<<i<<endl;
    return 0;
}
```

PROBLEM 5.1

Remember the encryption problem?

Now update your solution so that it deals with this issue:

- keys may be too large.

ARRAYS OF STRINGS

```
#include <iostream>
#include <cstdio>
using namespace std;
int main()
{
    char str_array[10][80];
    int a;
    for (a=0; a<10; a++)
     {
        cout<<"Enter array number "<<a+1<<endl;
        gets(str_array[a]);
    }
```

STRING CLASS

```
#include <iostream>
#include<string>
using namespace std;
int main()
{
    string str;
    getline(cin,str);
    cout<<str<<endl;
    return 0;
}
```

PROBLEM 5.2

Re-write your encryption programs using a string from the class string.

str.length() and str.size() return the number of bytes in str.

POINTERS

That’s an object that contains a memory address. In most cases, the location of another object such as a variable or function.

Type *[pointer name];
(eg: int *p;)

The type determines what type of data the pointer will be pointing to.

POINTER OPERATORS

&
That’s a unary operator that returns the memory address of its operand.
ptr = &total;
*
Returns the value of the variable located at the address specified by its operand.
val = *ptr;

```
#include<iostream>
using namespace std;
int main()
{
	int total=3200, val;
	int *ptr;
	ptr = &total;
	val = *ptr;
	cout<<"Total is: " <<val<<endl;
	cout<<"At address "<<ptr<<endl;
	return 0;
}
```

POINTER BASE TYPE

Using the correct base type allows transferring the right number of bytes for an assignment.

```
int *p;
double f;
p=&f; //ERROR
```

POINTER BASE TYPE II

```
//This program won't work as expected (but it will compile!)
#include <iostream>
using namespace std;
int main()
{
    double x, y;
    int *p;
    x=123.23;
    p = (int *) &x;
    y=*p;
    cout<<y<<endl;
    return 0;
}
```

ASSIGNING VALUES THROUGH A POINTER

```
#include <iostream>
using namespace std;
int main()
{
    int *p, num;
    p=&num;
    *p=100;
    cout<<num<<' ';
    (*p)++;
    cout<<num<<' ';
    (*p)--;
    cout<<num<<endl;
    return 0;
}
```

POINTER ARITHMETIC

```
int *ptr;
```

The above declares a pointer, let’s assume its located in the memory address 2000.

```
ptr++;
```

The address will be 2004, not 2001, why?

What about: ptr += 3?

```
#include <iostream>
using namespace std;
int main()
{
      int *i, j[10];
      double *f, g[10];
      int x;
      i=j;
      f=g;
      for (x=0; x<10; x++)
            cout<<i+x<<' '<<f+x<<endl;
      return 0;
}
```

PROBLEM 5.2

A better instalment calculation

Write a program that calculates the repayments of up to 50 loans. Each loan may have different parameters.

At the end your program should show a list of customer names and the value of their monthly payment.

$$Payment = \frac{IntRate * \frac{Principal}{PayPerYear}}{1 - \left(\left(\frac{IntRate}{PayPerYear}\right) + 1\right)^{-(PayPerYear * NYears)}}$$