String



- A string is actually a character array.
 - You can use it like a regular array of characters.
 - However, it has also some unique features that make string processing easy.

Initializing Strings



- Instead of initializing the elements of a string one-by-one, you can initialize it using a string.
- Eg:

Example



```
#include <iostream>
#include <string>
using namespace std;
int main ()
  char question1[] = "What is your name? ";
  char question2[] = "Where do you live? ";
  char answer1[80];
  char answer2[100];
  cout << question1;</pre>
  cin >> answer1;
  cout << question2;</pre>
  cin >> answer2;
  cout << "Hello, " << answer1;</pre>
  cout << " from " << answer2 << "!\n";
  return 0;
```

String Functions



- You can find several string functions in string.h.
 - strlen(), strcpy(), strcat(), strcmp()





- int strlen(char *st)
 - Returns the length of its string parameter (excluding null character).

• Assignment: Implement strlen() yourself.



strlen()

 Then, we can rewrite the lower-to-uppercase conversion as follows:

```
/* Convert lowercase to uppercase */
for (j=0; j<strlen(st); j++)
  if (('a'<=st[j]) && (st[j]<='z'))
     st[j] += 'A'-'a';
     Which one is better?</pre>
```



strcpy()

 If you want to copy the contents of a string variable to another, simple assignment does not work!

You have to copy the characters one-by-one.

There is a specific function that does this.

strcpy()



- char *strcpy(char *dest, char *source)
 - Copies all characters in source into dest.
 - Of course terminates dest with null char.
 - Returns starting address of dest.

Assignment: Implement strcpy() yourself.





```
char st1[5]="abdef", st2[5]="xyz";
strcpy(st1,st2);
st1[2]='M';
st2[3]='N';
printf("<st1:%s>\n",st1);
printf("<st2:%s>\n",st2);
```

What is the output?





- If you want to attach two strings, use strcat().
- char *strcat(char *dest, char *source)
 - Attaches source to the tail of dest.
 - Chars in dest are not lost.
 - Returns starting address of dest.

Assignment: Do it yourself.

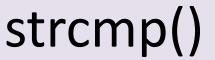


strcat()

 Write a function that reads a name from the input, prepends it with "Hello", and updates its parameter to contain this greeting string. (You may assume the caller passes a parameter that is large enough.)

```
void greet(char g_st[])
{    char name[20];
    scanf("%s", name);
    strcpy(g_st,"Hello ");
    strcat(g_st, name);
}
```

```
Why didn't we simply write g_st=strcat("Hello ",name);
```





- You may also check if the lexicographical ordering of two strings.
- int strcmp(char *st1, char *st2)
 - Returns < 0 if st1 comes before st2.
 - Returns 0 if st1 is identical to st2.
 - Returns > 0 if st1 comes after st2.

Assignment: Do it yourself.



Safe Operation

 As we discussed before, string functions are not safe in general since the size of the string is not controlled (everything depends on the occurrence of the null character).





- A solution is the use of safer functions: strncpy(), strncat(), strncmp()
 - -strncpy(dest,src,n)
 - Copy at most n characters of src to dest.
 - -strncat(dest,src,n)
 - Concatenate at most n characters of src to dest.
 - -strncmp(dest,src,n)
 - Compare at most n characters of dest.