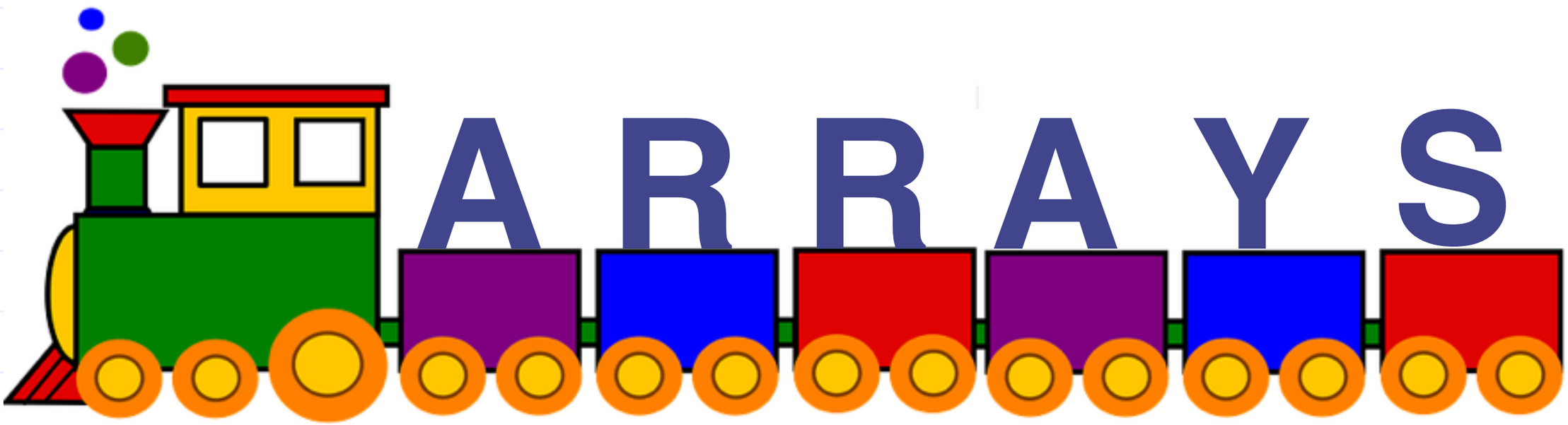
ESC101: Introduction to Computing



Array example: print backwards

Problem:

Define a character array of size 100 (upper limit) and read the input character by character and store in the array until either

- 100 characters are read or
- EOF (End Of File) is encountered

Now print the characters backwards from the array.

Example Input 1

```
Me or
Moo
```

Output 1

```
ooM
ro eM
```

Example Input 2

```
Eena Meena Dika
```

Output 2

```
akiD aneeM aneE
```

Read and print in reverse

1. We will design the program in a top down fashion, using the main() function.
2. There will be two parts to main: read_into_array and print_reverse.
3. read_into_array will read the input character-by-character up to 100 characters or until the end of input.
4. print_reverse will print the characters in reverse.

Overall design

```
int main() {
        char s[100]; /* to hold the input */
        /* read_into_array */
        /* print_reverse */
        return 0;
}
```

Let us design the program fragment read_into_array.

Keep the following variables:

1. int count to count the number of characters read so far.
2. int ch to read the next character using getchar().

Note that getchar() has prototype **int getchar()**
since getchar() returns all the 256 characters and the
integer EOF

```
int ch;
int count = 0;
read the next character into ch using getchar();
while (ch is not EOF AND count < 100) {
        s[count] = ch;
        count = count + 1;
        read the next character into ch using getchar();
}
```

An initial design (pseudo-code)

```
int ch;
int count = 0;
read the next character into ch using getchar();
while (ch is not EOF AND count < 100) {
        s[count] = ch;
        count = count + 1;
        read the next character into ch using getchar();
}
```

initial design
pseudo-code

```
int ch;
int count = 0;
ch = getchar();
while ( ch != EOF && count < 100) {
        s[count] = ch;
        count = count + 1;
        ch = getchar();
}
```

Translating the read_into_array pseudo-code into code.

Overall design

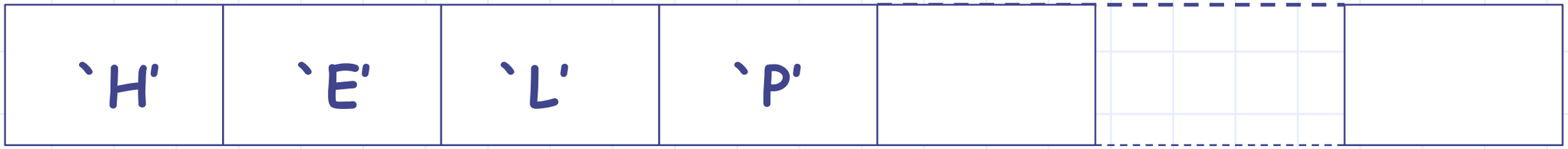
```
int main() {
    char s[100];
 /* read_into_array */
 /* print_reverse */
    return 0;
}
```

Now let us design the code fragment **print_reverse**

Suppose input is

HELP<eof>

The array char s[100]

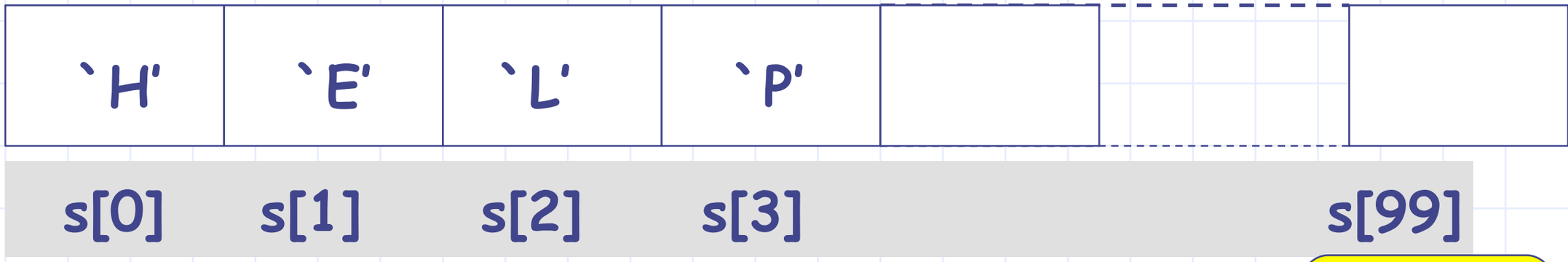



```
int i;
set i to the index of  last  character read.

while (i >= 0) {
    print s[i]
    i = i-1;       /* shift array index one to  left */
}
```

PSEUDO CODE

The array char s[100]




```
int i;
set i to index of the last character read.

while (i >= 0) {
    print s[i]
    i = i-1;
}
```

PSEUDO CODE

Translating pseudo code to C code: print_reverse

```
int i;
i = count-1;

while (i >=0) {
        putchar(s[i]);
        i=i-1;
}
```

Code for printing characters read in array in reverse

Putting it together

Overall design

```
int main() {
        char s[100];
        /* read_into_array */
        /* print_reverse */
        return 0;
}
```

The code fragments we have written so far.

```
int count = 0;
int ch;
ch = getchar();
while ( ch != EOF && count < 100) {
        s[count] = ch;
        count = count + 1;
        ch = getchar();
```

read_into_array code.

```
int i;
i = count-1;
while (i >=0) {
        putchar(s[i]);
        i=i-1;
}
```

print_reverse code

```
#include <stdio.h>
int main() {
      char s[100];
      int count = 0;
      int ch;
      int i;

      ch = getchar();
      while ( ch != EOF && count < 100) {
            s[count] = ch;
            count = count + 1;
            ch = getchar();
      }

      i = count-1;
      while (i >=0) {
            putchar(s[i]);
            i=i-1;
      }

      return 0;
}
```

```
/* the array of 100 char */
/* counts number of input chars read */
/* current character read  */
/* index for printing array backwards */
```

```
/*read_into_array */
```

```
/*print_in_reverse */
```

```c
#include <stdio.h>
int main() {
    char s[100];
    int count = 0;
    int ch;
    int i;
    while ( (ch=getchar()) != EOF &&
                    count < 100 )
    {       s[count] = ch;
            count = count + 1;
    }
    i = count-1;
    while (i >=0) {
            putchar(s[i]);
            i=i-1;
    }
    return 0;
}
```

/*read_into_array */

/*print_in_reverse */

Practice Problem

- Write a program to read in two character arrays. Each character array ends with a ‘,’ character or an EOF character. The output should be the joined characters

- Input: There was a kit,ten who liked milk
- Output: There was a kitten who liked milk

Solution for Practice problem

```
#include <stdio.h>
int main()
{
   char a[100], b[100], c[100];
   int ch, acnt=0, bcnt=0,i=0;

   return 0;
}
```

Solution for Practice problem

```
#include <stdio.h>
int main()
{
   char a[100], b[100], c[100];
   int ch, acnt=0, bcnt=0,i=0;
   ch=getchar();
   while( ch !=',' && ch !=EOF && acnt<100)
   {
      a[acnt] = ch;     acnt++;
      ch = getchar();
   }

return 0;
}
```

Solution for Practice problem

```
#include <stdio.h>
int main()
{
   char a[100], b[100], c[100];
   int ch, acnt=0, bcnt=0,i=0;
   ch=getchar();
   while( ch !=',' && ch !=EOF && acnt<100)
   {
      a[acnt] = ch;     acnt++;
      ch = getchar();
   }
   ch = getchar();
   while( ch !=',' && ch!= EOF && bcnt<100)
   {
      b[bcnt] = ch;     bcnt++;
      ch = getchar();
   }
return 0;
}
```

Solution for Practice problem

```
#include <stdio.h>
int main()
{
   char a[100], b[100], c[100];
   int ch, acnt=0, bcnt=0,i=0;
   ch=getchar();
   while( ch !=',' && ch !=EOF && acnt<100)
   {
      a[acnt] = ch;     acnt++;
      ch = getchar();
   }
   ch = getchar();
   while( ch !=',' && ch!= EOF && bcnt<100)
   {
      b[bcnt] = ch;     bcnt++;
      ch = getchar();
   }
   for(i=0; i<acnt; i++)
      c[i] = a[i];
 return 0;
}
```

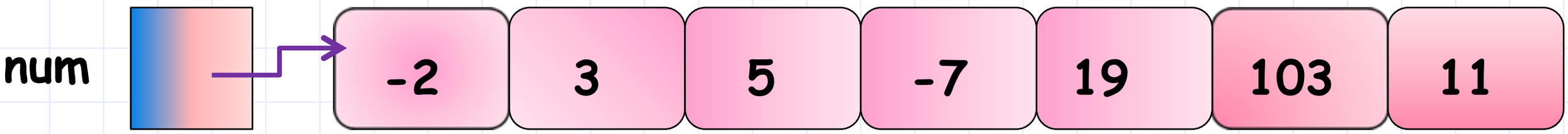
Solution for Practice problem

```
#include <stdio.h>
int main()
{
   char a[100], b[100], c[100];
   int ch, acnt=0, bcnt=0,i=0;
   ch=getchar();
   while( ch !=',' && ch !=EOF && acnt<100)
   {
      a[acnt] = ch;     acnt++;
      ch = getchar();
   }
   ch = getchar();
   while( ch !=',' && ch!= EOF && bcnt<100)
   {
      b[bcnt] = ch;     bcnt++;
      ch = getchar();
   }
   for(i=0; i<acnt; i++)
      c[i] = a[i];
   for(i=0; i<bcnt; i++)
      c[i+acnt] = b[i];
  return 0;
}
```

Solution for Practice problem

```
#include <stdio.h>
int main()
{
   char a[100], b[100], c[100];
   int ch, acnt=0, bcnt=0,i=0;
   ch=getchar();
   while( ch !=',' && ch !=EOF && acnt<100)
   {
      a[acnt] = ch;     acnt++;
      ch = getchar();
   }
   ch = getchar();
   while( ch !=',' && ch!= EOF && bcnt<100)
   {
      b[bcnt] = ch;     bcnt++;
      ch = getchar();
   }
   for(i=0; i<acnt; i++)
      c[i] = a[i];
   for(i=0; i<bcnt; i++)
      c[i+acnt] = b[i];
   for(i=0; i<acnt+bcnt; i++)
      putchar(c[i]);
   return 0;
}
```

How can we create an int array num[] and initialize it to:



Method 1

```
int  num[] = {-2,3,5,-7,19, 103, 11};
```

1. Initial values are placed within curly braces separated by commas.
2. The size of the array need not be specified. It is set to the number of initial values provided.
3. Array elements are assigned in sequence in the index order. First constant is assigned to array element [0], second constant to [1], etc..

Method

```
int  num[7] = {-2,3,5, -7, 19, 103, 11};
```

Specify the array size. size must be at least equal to the number of initialized values. Array elements assigned in index order.

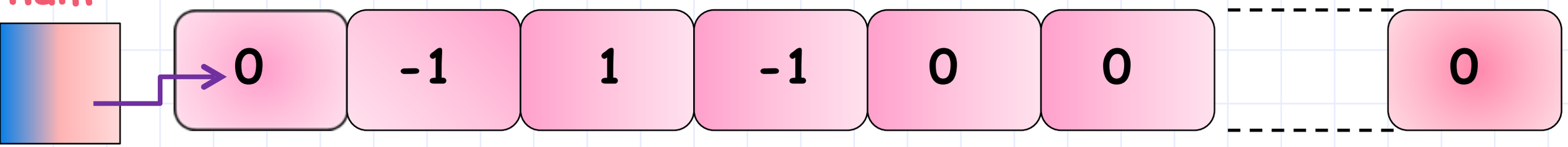
Recommended method: array size determined from the number of initialization values.

```
int  num[] = {-2,3,5,-7,19,103,11};
```

Is this correct?

```
int num[100] ={0,-1,1,-1};
```

YES Creates num as an array of size 100. First 4 entries are initialized as given. num[4] … num[99] are set to 0.



Is this correct? **NO! it won't compile!**

```
int  num[6] = {-2,3,5,-7,19,103,11};
```

Why?

1. **num is declared to be an int array of size 6 but 7 values have been initialized.**
2. **Number of initial values must be less than equal to the size specified.**

Initialization values could be constants or constant expressions. Constant expressions are expressions built out of constants.

```
int num[] = { 109, 'A', 7*25*1023 +'1' };
```

Type of each initialization constant should be promotable/demote-able to array element type.

E.g.,

```
int num[] =  { 1.09, 'A',25.05};
```

Float constants 1.09 and 25.05 downgraded to int

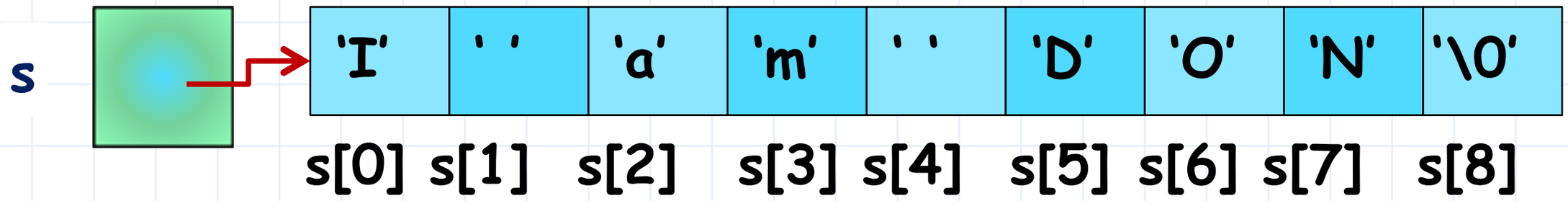
Would this work?

```
int curr = 5;
int num[] = { 2, curr*curr+5};
```

YES. C allows constant expressions AND simple expressions for initialization values. "Simple" is compiler dependent.

Character array initialization

Character arrays may be initialized like arrays of any other type. Suppose we want the following char array.



We can write:

```
s[]={'I',' ','a',' m',' ','D','O','N','\0' };
```

BUT! C allows us to define string constants. We can also write:

```
s[]  ="I am DON";
```

1. **"I am DON" is a string constant. The '\0' character (also called NULL char) is automatically added to the end.**

2. **Strings constants in C are specified by enclosing in double quotes e.g. "I am a string".**

Reading a String (scanf)

- Placeholder: %s
- Argument: Name of character array.
- No & sign before character array name. (?)
- Input taken in a manner similar to numeric input.
- With %s, scanf skips whitespaces.
  - There are three basic whitespace characters in C : space, newline ('\n') and tab ('\t').
  - Any combination of the three basic whitespace characters is a whitespace.

Reading a String (scanf)

- Starts with the first non-whitespace character.
- Copies the characters into successive memory cells of the character array variable.
- When a whitespace character is reached, scanning stops.
- scanf places the null character at the end of the string in the array variable.

```
#include <stdio.h>

int main() {
 char str1[20], str2[20];

 scanf("%s",str1);
 scanf("%s",str2);

 printf("%s + %s\n", str1,
str2);

 return 0;
}
```

INPUT
IIT Kanpur

OUTPUT
IIT + Kanpur

INPUT
I am DON

OUTPUT
I + am

NULL character '\0'

- ASCII value 0.
- Marks the end of the string.
- C needs this to be present in every string in order to differentiate between a character array and a string.
- Size of char array holding the string ≥ 1 + length of string
  - Buffer overflow otherwise!

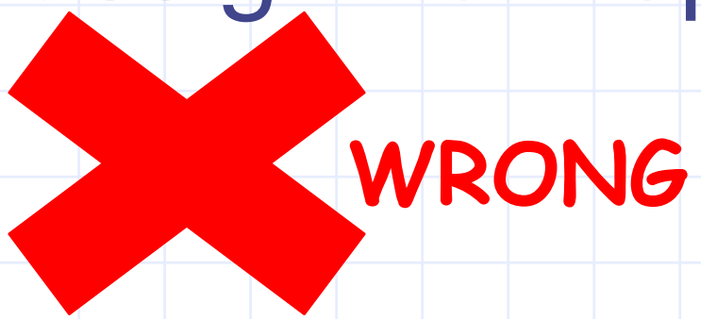
NULL character ‘\0’

- What happens if no '\0' is kept at the end of string?
  - ‘\0’ is used to detect end of string, for example in printf(“%s”, str).
  - Without ‘\0’, such functions will keep reading array elements beyond the array bound (out of bound access).
  - We can get an incorrect result or a Runtime Error.

Copying a String

- We can not copy content of one string variable to other using assignment operator

```
char str1[] = "Hello";
char str2[] = str1;
```



Array type is not assignable.

C Pointers needed!

  - This is true for any array variable.
  - Error: Array initializer must be a list or a string.
- We need to do element-wise copying

String Copy

- Two char arrays src[]and dest[].
- Copy contents of src into dest.
- We assume that dest is declared with size at least as large as src.
- Note the use of '\0' for loop termination

```
// declare and initialise char src[]
// declare char dest[]
   int i;
   for (i = 0; src[i] != '\0'; i++){
         dest[i] = src[i];
}
   dest[i] = '\0';
```