Printing strings

We have used string constants many times. Can you recall?

printf and scanf: the first argument is always a string.
1. printf("The value is %d\n", value);
2. scanf("%d", &value);

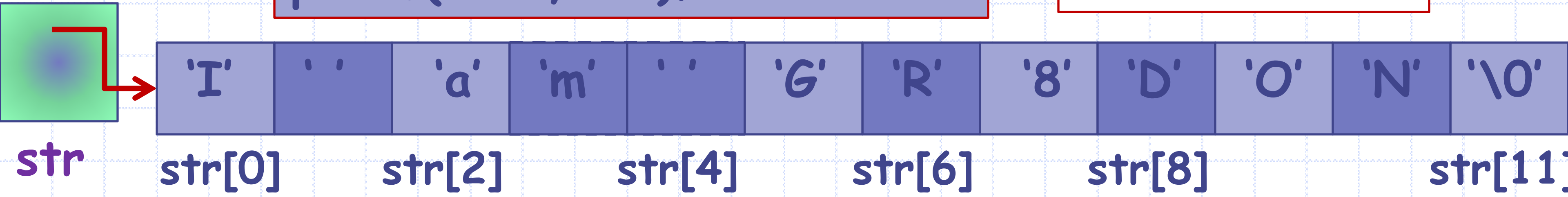
Strings are printed using %s option.

E.g. 1
```
printf("%s", "I am DON");
```

Output
```
I am DON
```

E.g. 2
```
char str[]="I am GR8DON";
printf("%s", str);
```

Output
```
I am GR8DON
```



State of memory after definition of str in E.g. 2. Note the NULL char added in the end.

This NULL char is not printed.

Strings

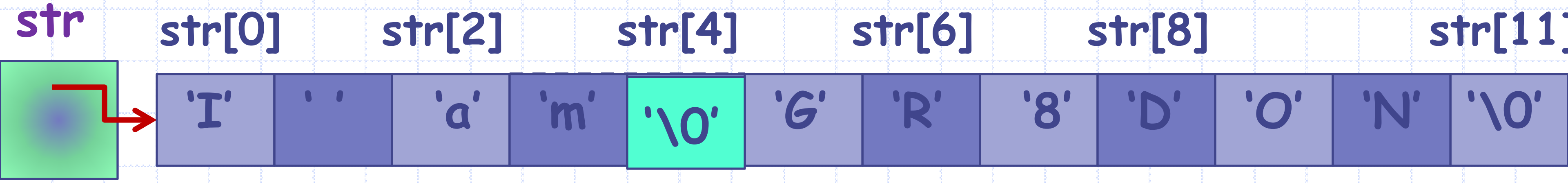
Consider the fragment.

```
char str[]="I am GR8DON";
str[4] = '\0';
printf("%s", str);
```

This defines a constant string, i.e., character array terminated by, but not including, '\0'.

What is printed?

Let us trace the memory state of str[].



Output

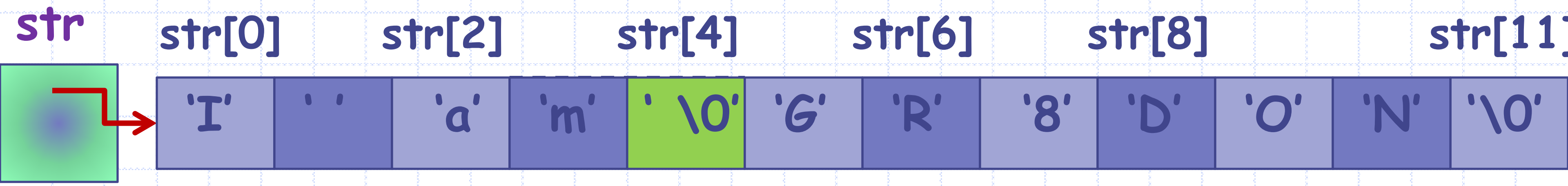
I am

1. A string is a sequence of characters terminated by '\0'. This '\0' is not part of the string.

2. There may be non-null characters after the first occurrence of '\0' in str[]. They are not part of the string str[] and don't get printed by printf("%s", str);

So did we lose the chars after the first '\0' ? Where did they go?

Of course not, they remain right where they were. They were not printed because we used %s in printf. Let's take a look.



```
char str[]="I am GR8DON";
str[4]='\0';
printf("%s", str);
```

Output

```
I am
```

```
int i;
for (i=0; i < 11; i++) {
        putchar(str[i]);
}
```

Output

```
I amGR8DON
```

The character '\0' may be printed differently on screen depending on terminal settings.

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
```

Why is there no & when we read character array?

Remember parameter passing?

- A simple variable can not be modified from inside a function call (Recall swap() function)
- However, Arrays can be modified from inside a function call (Recall read_into_array() function)
- Similarly, scanf can also “modify” arrays directly. Since string is just an array of chars, & is not required.
- More on this when we do pointers

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.

Reading a line as an input

- scanf, when used with the %s placeholder, reads a block of non-whitespace characters as a string.
- What if we want to read a line as a string?
- We will define our own function to read a line.
- **EXERCISE:** Take as input a line (that ends with the newline character) into a character array as a string.

```
#include <stdio.h>
// read a line into str, return length
int read_line(char str[]) {
   int c, i=0;
   c = getchar();
   while (c != ‘\n’ && c != EOF) {
      str[i] = c;
      c = getchar();
      i++;
   }
   str[i] = ‘\0’; // we want a string!
   return i; // i is the length of the string
}
```

DANGER!

Buffer overflow possible

```
#include <stdio.h>
// read a line into str, return length.
//   maximum allowed length is limit
int read_line(char str[], int limit ) {
    int c, i=0;
    c = getchar();
    while (c != '\n' && c != EOF) {
        str[i] = c;
        c = getchar();
        i++;
        if (i == limit-1) break;
    }
    str[i] = '\0'; // we want a string!
    return i; // i is the length of the string
}
```

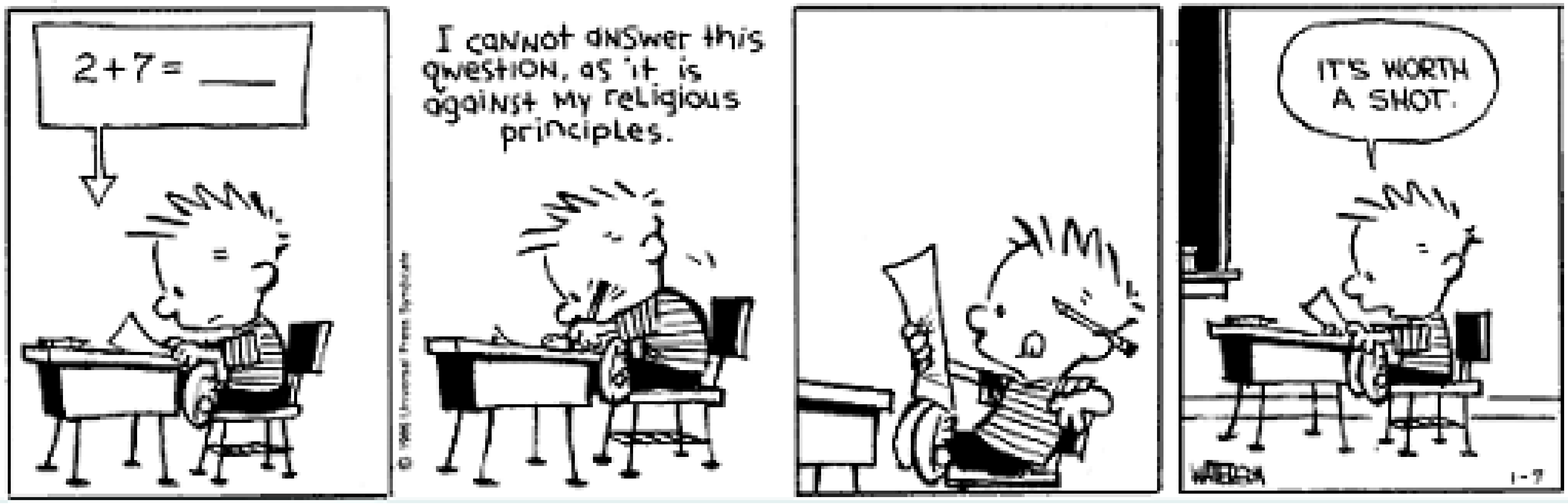
Safer version!

ESC101: Introduction to Computing

Lab Exam &

Mid semester Exam

Ack: Bill Watterson for Calvin & Hobbes images


2+7= ___
I cannot answer this question, as it is against my religious principles.
IT'S WORTH A SHOT.

Tips for Lab Exam

- Read all instructions and the questions carefully.
- Approaching a problem:
  - Don't try to write the entire program in one attempt.
  - Write short segments of code and keep compiling whenever possible to ensure correctness.
  - Use functions to divide your program into smaller components.
  - Write functions to perform ONE task at a time. Combine the tasks in main.

Tips for Lab Exam

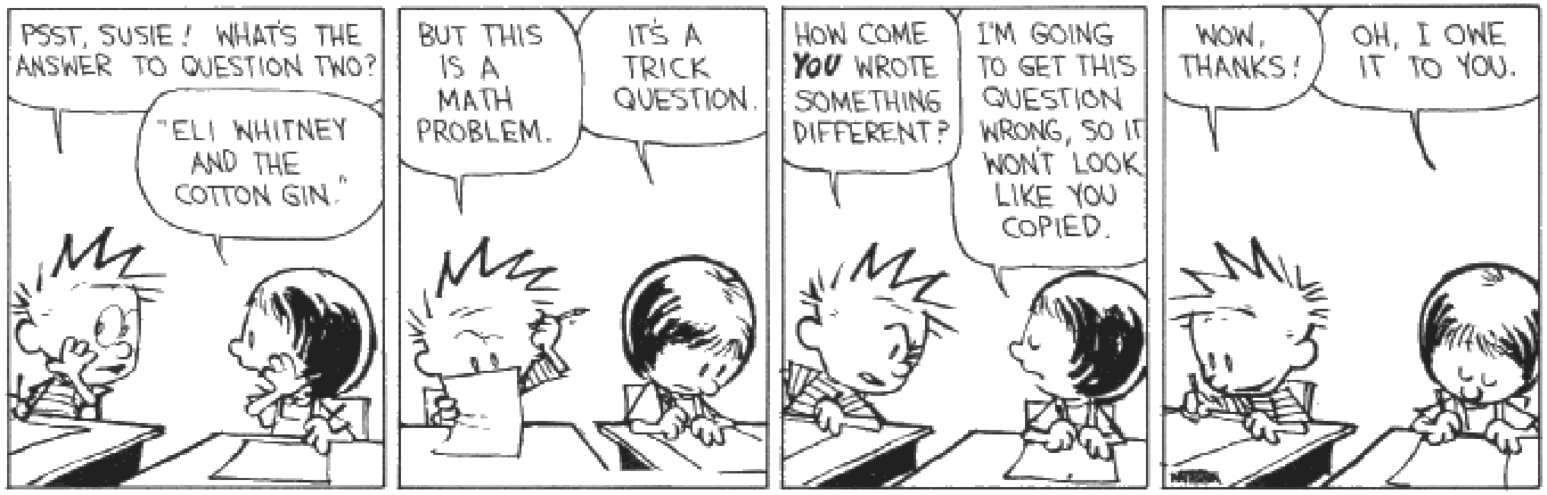
- If your program is not working
  - insert dummy printf statements to test where it is going wrong.
    - Specially test input/output of the functions you write.
  - make a variable table (especially when using loops).
  - Debug ONE iteration of a loop at a time.
- Important: Write comments to explain your code.
- Give meaningful variable names.
- Do not rely solely on the test cases provided by us.

Conduct for Lab Exam

- Lab exam will be conducted through esc101.cse.iitk.ac.in
- The network will be ON during the exam
- Closed book exam
  - You are **not** allowed to access any other site during the exam
  - Do **not** check emails (@iitk, @gmail, ...)
  - Mobiles are **not** allowed on person (even if switched off)
- ITS stores your *keystrokes and program development steps!*

Conduct for Lab Exam

- We will monitor system usage
  - Tracking IP used for submission and incoming and outgoing network traffic
  - Manually and automatically
- Zero tolerance policy for Cheating
  - F grade AND Report to DoAA to be filed in student's record


PSST, SUSIE! WHAT'S THE ANSWER TO QUESTION TWO?
"ELI WHITNEY AND THE COTTON GIN."
BUT THIS IS A MATH PROBLEM.
IT'S A TRICK QUESTION.
HOW COME YOU WROTE SOMETHING DIFFERENT?
I'M GOING TO GET THIS QUESTION WRONG, SO IT WON'T LOOK LIKE YOU COPIED.
WOW, THANKS!
OH, I OWE IT TO YOU.

Tips for Midsem Exam

- Read the instructions.
- Read the question carefully.
- In questions where you are required to give the output of a program, understand what the code is doing.
  - ✓ Check for tricky constructs

Mr. Jones lives 50 miles away from you. You both leave home at 5:00 and drive toward each other.
Mr. Jones travels at 35 mph., and you drive at 40 mph. At what time will you pass Mr. Jones on the road?
GIVEN THE TRAFFIC AROUND HERE AT 5:00, WHO KNOWS?
I ALWAYS CATCH THESE TRICK QUESTIONS.

Tips for Midsem Exam

- Questions where you need to complete a partially filled program:
  - ✓ First understand the problem.
  - ✓ Then try to understand the given code.
  - ✓ Complete the program.
  - ✓ Check whether the completed program is behaving as it should.
- Finally check whether you have answered all questions and verify your answers.
- Practice!

LAB Exam duration: **2:15 PM - 5:00 PM**.

On Tuesday, 8th Sep @ 2:15 PM

B1,B2,B3 (Monday Lab Batch):

Reporting at **Computer Center Lab CC L2 / 3**

B4,B5,B6 (Tuesday Lab Batch):

Reporting at **New Core Labs**

On Wednesday, 9th Sep @ 2:15 PM

B7,B8,B9 (Thursday Lab Batch):

Reporting at **Computer Center Lab CC L2 / 3**

B10,B11,B12 (Wednesday Lab Batch):

Reporting at **New Core Labs**

Sample Question: Euclid's Number

- The n-th *primorial* number, pn is the product of the first n primes.
  - E.g. p1 = 2, p2 = 6, p3 = 30, and so on.
- The n-th *Euclid* number, En is defined as En = pn + 1.
  - E.g. E1 = 3, E2 = 7, E3 = 31, and so on.
- Question: Write a program that displays the first n Euclid numbers and states for each number whether it is prime or composite.

Solving for Euclid’s Numbers

- We can use the top down approach to solve the problem.
- Assume a function to display Euclid numbers (with prime/composite-ness) and use it in main.

```
#include <stdio.h>
void display_euclid(int n); // decl. only
int main(){
    int val;
    scanf("%d",&val);
    display_euclid(val);
    return 0;
}
```

Now fill in the details of display_euclid

- May require more assumed functions!

```
int primorial(int n);
int is_prime(int n);

// display first n the Euclid number and whether they
// are prime or composite.
void display_euclid(int n){
    int i, en;

    for (i=1; i<=n; i++) {
        en = primorial(i)+1;
        if (is_prime(en) == 1)
            printf("%d: Prime\n", en);
        else
            printf("%d: Composite\n", en);
    }
}
```

- Keep filling the details of assumed functions
  - Till bottom out (where no more functions are assumed)

```c
// return the nth primorial no.
int primorial(int n)
{
    int count=0, i=2, prod=1;
    while (count<n){
        if (is_prime(i) == 1){
            prod = prod * i;
            count++;
        }
        i++;
    }
    return prod;
}
```

```c
// return 1 if n is prime,
// else return 0.
int is_prime(int n){
    int i;
    for (i=2; i*i<=n; i++){
        if (n%i == 0)
            return 0;
    }
    return 1;
}
```

```c
#include <stdio.h>
void display_euclid(int n);
int main(){
    int val;
    scanf("%d",&val);
    display_euclid(val);
    return 0;
}

int primorial(int n);
int is_prime(int n);
// display first n the euclid number ...
void display_euclid(int n){
   int i, en;
   for (i=1; i<=n; i++) {
       en = primorial(i)+1;
       if (is_prime(en) == 1)
           printf("%d: Prime\n",en);
       else
           printf("%d: Composite\n",en);
   }
}
```

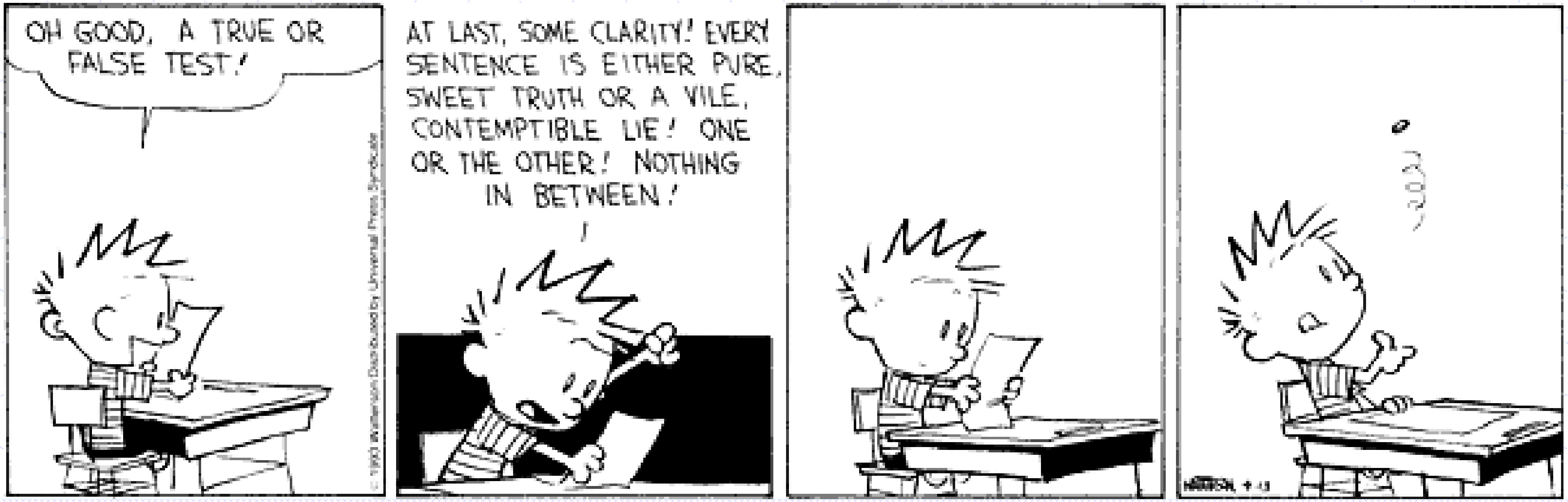
```c
// return the nth primorial no.
int primorial(int n)
{
    int count=0, i=2, prod=1;
    while (count<n){
        if (is_prime(i) == 1){
            prod = prod * i;
            count++;
        }
        i++;
    }
    return prod;
}

// return 1 if n is prime ...
int is_prime(int n){
    int i;
    for (i=2; i*i<=n; i++) {
        if (n%i == 0)
            return 0;
    }
    return 1;
}
```

Notes...

- Depending on the requirement, program can be made more efficient
  - use Sieve of Eratosthenes to compute primes
  - "*Memorization*" to compute primorial
    - p(n) = p(n-1) * N, where N is n-th prime
    - Use of array to store p(1) ... p(n)
  - Use of long to increase domain
    - Avoid overflow longer...
    - Need special tricks to hold numbers bigger than long

All the best …


OH GOOD, A TRUE OR FALSE TEST!
AT LAST, SOME CLARITY! EVERY SENTENCE IS EITHER PURE, SWEET TRUTH OR A VILE, CONTEMPTIBLE LIE! ONE OR THE OTHER! NOTHING IN BETWEEN!
© 1990 Watterson Distributed by Universal Press Syndicate