Lecture 3:
More C: Functions and Arrays

In the course thus far: C programs that do little calculations .
- Use Arithmetic Operations: +, *, /, %;..
- Use `for, while, if-else,...`

Reading for last week's material:
- *Survival Guide*: "Week 0" and "Week 1"
- Course Reader: Pages 1--12; 20--26

Be Forewarned: C is not designed for beginners.
ASK US LOTS OF QUESTIONS;
EXPERIMENT.

Another Example of Functions

```
#include <stdio.h>
float square(float);
float cube(float);
main()
{
  float a =2.718;

  printf("%f\n", cube(a));
}
float square(float b)
{
  return b*b;
}
float cube(float c)
{
  return c* square(c);
}
```

- Function declaration must precede its use.
- A function can call any other function (including itself!)
- The name of the function's argument has no special significance.

More about functions

- Most standard math functions are available in C

```
#include <stdio.h>
#include <math.h>
void main()
{
  float c=2.0;
  printf("%f\n", exp(c));
}
```

  Other available functions: `sqrt(x), log(x), pow(x,y), sin(x), ....`

- `printf` and `scanf` are also functions

- **dividing up large programs into small functions helps readability**

"Scope" of variables in function calls

What does the following program output?

```
#include <stdio.h>
void tester(int);
void main()
{
  int i=5;

  tester(i);
}

void tester(int j)
{
  printf("%d", i);
}
```

"Scope" (contd.)

What does the following program output?

```
#include <stdio.h>
void squarer(int i)
{
 i = i*i;
}

void main()
{
  int i=5;

  squarer(i);
  printf("%d", i);
}
```


Scope Rules

- caller function's variables are "invisible" to the callee function.

- "call by value" rule: the callee function gets only the *value* of the argument, not its address. (important: we will see an exception later)

Reason for these rules: **They prevent Chaos!**

Question: How to write programs that manipulate large amounts of data? (airline reservation system, my program for assigning grades, ...)

A Messy Grading Program :
150 variables, if-then-else with 150 cases..

Better: Use **Arrays.**

```
#include <stdio.h>
void main()
{ int i;
  int grades[150];

  for(i=1; i<150; i++)
     scanf("%d", &grades[i]);
  for(i=0; i<150; i++)
   {
        if (grades[i]> 80)
           printf("Student %d passes\n", i);
        else
           printf("Student %d fails\n", i);
    }
}
```


Arrays

How to declare:

```
char   a[50], b[80];
float  c[20];
```

Notes:

- Array size must be known at compile time.
  This program generates a compiler error.

```
void main()
 {
   int i=5;
   int a[i];
```

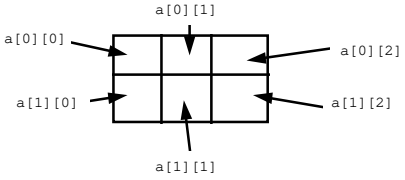
- Arrays begins at index 0
  (source of many off-by-one errors!)

Example: `int a[4];`
defines an array with elements
`a[0], a[1], a[2], a[3].`

Multidimensional Arrays

```
void main()
{
  int a[2][3];
```

defines an integer array a with 6 entries



Example 1: 150 students; 4 exams
(150 x 4 = 600 grades to keep track of)

```
int grades[150][4];
```

Example 2: Bitmap

```
int color[640][480];
for(i=0; i<640; i++)
 for(j=0; j<480; j++)
   color[i][j] = computed bit color
```

Symbolic Constants

```
#include <stdio.h>
#define NUM_STUDENTS 150
void main()
{
  int i;
  int grades[NUM_STUDENTS];

  for(i=0; i <NUM_STUDENTS; i++)
     scanf("%d", &grades[i]);
}
```

- **Allows "important"constants to receive a name**
- **Provides error-free way to change these constants**

Aside: Statements beginning with # are commands to the C *preprocessor.*

Passing arrays as arguments to functions

```
#include <stdio.h>
#define NUM_STUDENTS 150
void update_ grades(int a[])
void main()
{
  int grades[NUM_STUDENTS];
   .
   .
  update_grades(grades);
}
void update_inventory(int a[])
{
  for(i=0; i <NUM_STUDENTS;i++)
     {
        a[i] = 2*a[i];
         .
      }
}
```

Note: Arrays are not passed using call-by-value

Global Variable

A variable that is "visible" to all functions
(provides a way to get around limitations
imposed by scope rules)

```
#include <stdio.h>
#define NUM_STUDENTS 150
int funnygrades[NUM_STUDENTS];
void update_grades();
void main()
{
 funnygrades[0]=150;
 init_grades();
}
void init_grades()
{ int i;
    for(i=1; i < NUM_STUDENTS;i++)
     {
        funnygrades[i] =10;
     }
}
```