Lecture Overview:

Programming basics in C (*for a graphics course*):
- “Hello, world”
- Variables and data types
- Operators
- Flow control: decisions and loops
- Prototypes and Header files
- Functions
- Pointer basics
- Arrays
- I/O
- Dynamic memory allocation

Assignment #1 - Matrix algebra exercises
MAE 473-573: Lecture #3 - Introduction to C

Teaching Assistants:

1. Nidal A Al-Masoud    naa2@eng.buffalo.edu

Newsgroup: sunyab.mae.573
“Hello, world”

• The simplest C program
• Exhibits the basic elements of every C code:
  (header, function, brackets, semicolons)

```c
#include <stdio.h>

void main()
{
  printf("hello, world\n");
}
```
Variables, data types, strings

*Data types:*

- `int value = 1234;` (-32768 : 32768)
- `long int bigvalue = 123456789;` (-2147483648 : 2147483648)
- `unsigned int uint = 5678;` (0 : 65535)
- `float var = 45.678;` (e-38 : e+38)
- `double dvar = 376.889012` (e-308 : e+308)

*Strings:*

- `char ex_string[80];`
- `scanf("%s", ex_string);`
- `printf("You entered %s", ex_string);`
Operators

- Arithmetic: +, -, *, /, %
- Comparison: >, >=, <, <=, ==, !=
- Conditional: &&, ||
- Increment/decrement: ++, --
- Assignment: =, +=, -=, *=, /=, %=, …

Example: i = i + 6;
          i += 6;
Flow control: decisions and loops

If-else

If(x > 0)
  y = 2;
else
  y = 4;

Else-if

If(x > 0)
  y = 2;
else if(x < 0)
  y = 4;
else
  y = x;

Switch

int choice;
switch(choice)
{
  case 1:
    printf("case = 1\n");
  case 2:
    printf("case = 2\n");
}

For

for(i = 0; i < 15; i++)
{
  x = x*pow(x, i);
}

Do-while

i = 10;
do
{
  i--;
} while (i < 5)

goto
Prototypes and Header files (*.h)

```c
#include <stdio.h>
#include <math.h>

int compute(int b);

void main()
{
    int i, b=2;
    printf("hello, world\n");
    i = compute(b);
}
```

- Other common headers: `<stdlib.h>`, `<stddef.h>`
- Prototype: after include, before main()
Functions
• “Divide and conquer” - the heart of top-down programming
• Partition you main code into “subtasks”
• Makes for a cleaner, elegant, more efficient code

```c
int addit(int a, int b)
{
    return a+b;
}

void printAmessage(void)
{
    printf("This function prints a message.\n");
}

double myNumber(void)
{
    double x = 31.2;
    double y = -97.3;
    return x*y/23.1;
}

void swap(int a, int b)
{
    int temp;
    temp = a;
    a = b;
    b = temp;
    /* Is this doing what we want it to? */
}
```

• Functions - call by \textit{value}
• Variables in function - local!!
Pointer basics

DEF: Represents the address (location) of information stored in memory....A memory address that “points” to another value.

```c
int i; /* An integer variable i. */
int *p; /* A pointer p to an integer value. */
p = &i; /* Assigns the address of integer I to pointer p. */
/* p now “points” to the memory LOCATION where i’s value is stored. */

To print the value of i:    printf("%d", i);
    printf("%d", *p);
```
Pointer basics
The corrected “swap” function - use pointers!

```c
#include <stdio.h>
void swap(int *a, int *b);

void main()
{
    int i = 5, j = 9;
    swap(&i, &j);    /* Pass the address of the variables. */
}

void swap(int *a, int *b)
{
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
}
```

• Using a pointer to attain its addressed data: *dereferencing* the pointer
Arrays

• Useful to represent vectors and matrices
• Widely used in computer graphics!
• Array indexing starts from 0, not 1!!

• char name[32]; /* String which has a maximum of 32 characters. */
  j = 6;
  sprintf(name, “Kevin is %1d feet tall\n”, j);
• int scores[10]; /* integer array has 10 elements: scores[0], scores[1], … */
• float maxx[5];
  for(j = 0; j < 5; j++)
  {
    maxx[j] = j-1;
  }
• #define MAX 5 /* Symbolic constant - top of program! */
  double vals[MAX] = {0, 1, 2, 3, 4}
• int squared[5][4];
• float cubic[3][4][5];
MAE 473-573: Lecture #3 - Introduction to C

I/O
(Requires the #include <stdio.h> header file)

Formatting:  
%\texttt{d} - integer  
%\texttt{l}d - long integer  
%\texttt{f} - floating point  
%\texttt{l}f - double precision  
%\texttt{s} - character string

Screen I/O:
printf: printf(“Stiffness K[\texttt{d}][\texttt{d}] = %lf\backslash n”, i, j, k[i][j]);  
scanf: printf(“What is the integer value?\backslash n”);  
scanf(“\texttt{d}i\texttt{f}”, &i);  /* Use the address of the value for scanf/fscanf. */

File I/O:  
FILE *fp; /* File pointer for the file to be accessed. */  
fprintf fp = fopen(“output.dat”, “w”);  
fscanf for(j = 0; j < 3; j++)  
{  
    fprintf(fp, “%lf\backslash n”, k[j]);  
}  
fclose(fp);

fp = fopen(“input.dat”, “r”);  
for(j = 0; j < 3; j++)  
{  
    fscanf(fp, “%lf\backslash n”, &k[j]);  
}  
fclose(fp);
Dynamic memory allocation
- Useful for efficient array usage; **multi-dimensional arrays**
- Use “malloc” to dynamically allocate array space

- Declare the array:
  double *a;
- Allocate sufficient space for the array:
  a = (double *)malloc(numss * sizeof(double));
/* This is double array a[i], where i ranges from 0 to numss-1. */
- Use the array
  a[..] = ..;
- When done, free the memory for the array space:
  freemem(a);
• Understanding *indexing* is the real trick…

Now consider a 2D array, \( b[j][k] \):

\[ j: 0, \text{numss}-1 \]
\[ k: 0, \text{maxbv} - 1 \]

\( b = (\text{double} *)\text{malloc(\text{numss} \times \text{maxbv} \times \text{sizeof(double)})}; \)

/* Now, a 1-D array… */

Example: assume \( \text{numss}=3, \text{maxbv}=4 \) (array has 12 elements)
To access “\( b[1][3] \)” dynamically:

\( b[\text{maxbv} \times i + j] = \ldots; \)
Miscellaneous

Comments: Use "/*" to begin and "*/" to end a comment.
/* This is a comment in ANSI C. */

Global variables: Global to all functions. Use with caution!!
Declare prior to main() function.

Type conversion: Convert between data types.
var1 = (type) var2;
f = (float)i; /* f is a float; i is an int. */
i = (int)f;

UNIX compile/link/run:

cc -o exe filename1.c filename2.c -lm /* Always link to math lib! */
exe > output.dat /* screen dump to output file! */
Common Intrinsic math functions:

- Remember - use `<math.h>`, and link using “-lm”!!

\[
\begin{align*}
\sin(x), \cos(x), \tan(x), \ldots & \quad /* \text{Trig functions in radians}!!! */ \\
\exp(x) & \quad /* e^x */ \\
\log(x) & \quad /* \ln(x) */ \\
\log10(x) & \quad /* \log_{10}(x) - \text{base 10 logarithm} */ \\
pow(x, y) & \quad /* x^y - \text{exponentiation}! */ \\
sqrt(x) & \quad /* \text{square root} */ \\
floor(x) & \quad /* \text{integer portion of double } x */ \\
fabs(x) & \quad /* \text{absolute value of double } x */ \\
fmod(x, y) & \quad /* \text{modulus - remainder of } x/y */
\end{align*}
\]
References

• “The C Programming Language. 2nd edition”
  Brian Kernighan
  Dennis Ritchie
  Prentice Hall

• My advice: find a good “learn by example” book.
  Check out this link:
  http://users.powernet.co.uk/eton/clc/cbooks.html