CSC231—Bash Labs

Week #10, 11, 12 — Spring 2017

Introduction to C

Dominique Thiébaut
dthiebaut@smith.edu
Learning C in 4 Hours!

D. Thiebaut
- Dennis Ritchie
- 1969 to 1973
- AT&T Bell Labs
- Close to Assembly
- Unix
- Standard
- Many languages based on C. (C++, Obj. C, C#)
- Many influenced by C (Java, Python, Perl)
C Lacks...

• Exceptions
• Garbage collection
• OOP
• Polymorphism
• But...
C Lacks...

• Exceptions
• Garbage collection
• OOP
• Polymorphism
• But... it is usually faster!
Good Reference

Hello World!

• Library
• Strings
• Block-structured language
• main()

```c
#include <stdio.h>

void main() {
    printf("\nHello World\n");
}
```
Hello World!

- Library
- Strings
- Block-structured language
- main()

```c
#include <stdio.h>

void main() {
    printf("\nHello World\n");
}
```
Compiling on Aurora

• gcc Gnu compiler
• man gcc for help

[~/handout]$ gcc hello.c
[~/handout]$ ./a.out

Hello World

[~/handout]$ gcc -o hello hello.c
[~/handout]$ ./hello

Hello World
[~/handout]$ ls -l
total 28
-rwx------ 1 352a 352a 6583 Oct  6 16:41 a.out*
-rwx------ 1 352a 352a 6583 Oct  6 16:48 hello*
-rw------- 1 352a 352a   66 Oct  6 16:41 hello.c
-rw------- 1 352a 352a   67 Oct  6 16:39 hello.c~
• Write your own Hello World! program

• Make it print something like:

```
************
* C Rocks! *
************
```
The GNU C Reference Manual

Table of Contents

- The GNU C Reference Manual
- Preface
  - Credits
- 1 Lexical Elements
  - 1.1 Identifiers
  - 1.2 Keywords
  - 1.3 Constants
    - 1.3.1 Integer Constants
    - 1.3.2 Character Constants
    - 1.3.3 Real Number Constants
    - 1.3.4 String Constants
  - 1.4 Operators
  - 1.5 Separators
  - 1.6 White Space
- 2 Data Types
  - 2.1 Primitive Data Types
    - 2.1.1 Integer Types
    - 2.1.2 Real Number Types
    - 2.1.3 Complex Number Types
• `printf( "string with %-operators", list of vars);`
  • `%d` int
  • `%f` float
  • `%s` string
Variables

• Simple types

• No strings!

• No booleans (only 0 for false and !0 for true)

• No classes, no objects!

int    -> integer variable
short  -> short integer
long   -> long integer
float  -> single precision real (floating point) variable
double -> double precision real (floating point) variable
char   -> character variable (single byte)
/*
programName.c
author

This is the header */
#include <stdio.h>
#include <string.h>

void main() {
    // line comment
    print( "Hello!\n" )
}
#include <stdio.h>
#include <string.h>

void main() {
    char hello[] = "hello"
    char world[] = "world!"
    char sentence[100] = "";

    strcpy( sentence, hello ); // sentence <- "hello"
    strcat( sentence, " " ); // sentence <- "hello "
    strcat( sentence, world ); // sentence <- "hello world!"

    printf( "sentence = %s\n", sentence );
}

[/handout]$ gcc strings2.c
[/handout]$ a.out

sentence = hello world!
[/handout]
Strings end with ‘\0’

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

void main() {
    char sentence[100] = "Hello world!";

    printf( "sentence = %s\n", sentence );
    sentence[5] = '\0';
    printf( "sentence = %s\n", sentence );
}
```

```bash
~/handout$ a.out
sentence = Hello world!
sentence = Hello
```

• make the program store the sum of a and b into c, and then print your full name and the value in c. Also, make it output the number of characters in your full name; it must count the number of chars using a string function (use strlen).
```c
#include <stdio.h>

void main() {
    int i;
    int sum = 0;

    // compute the sum of all the numbers from 1 to 100
    for ( i=1; i<=100; i++ ) {
        sum += i;
    }

    printf( "\nsum = %d\n\n", sum );
}
```
#include <stdio.h>

void main() {
    int i;
    int sum = 0;

    // compute the sum of all the numbers from 1 to 100
    i = 1;
    while ( i<=100 ) {
        sum += i;    // could have also used i++
        i += 1;
    }

    printf( "\nsum = %d\n\n", sum );
}
#include <stdio.h>

void main() {

    while ( 1 ) {
        printf( "hello!\n" );
    }
}

#include <stdio.h>

void main() {

    for ( ;; ) {
        printf( "hello!\n" );
    }
}
Exercise

• Write a program that displays your full name underlined (line of dashes below). The underline length must be computed and the number of characters equal to the number of characters in your full name.

Hints: strlen() returns # of chars

man strlen
Exercise

• Write a program that displays a triangle of $N$ lines of stars:

*  
**  
***  
****  
*****  

Homework!
Symbolic Constants

#include <stdio.h>

#define NAME "Mickey"
#define HEIGHT 5
#define YEARBORN 1928

void main() {
    printf( "%s is %d inches high, and was created in %d\n\n", 
            NAME, HEIGHT, YEARBORN );
}

#include <stdio.h>

#define NAME      "Mickey"
#define HEIGHT    5
#define YEARBORN  1928

void main() {
    printf( "%s is %d inches high, and was created in %d\n\n", "Mickey", 5, 1928 );
}
#include <stdio.h>

void main() {
    int a = 5;
    int b = 3;
    int c = 7;

    if ( a <= b && a <= c )
        printf( "%d is the smallest\n\n", a );
    else if ( b <= a && b <= c )
        printf( "%d is the smallest\n\n", b );
    else
        printf( "%d is the smallest\n\n", c );
}
# Conditionals

<table>
<thead>
<tr>
<th>&amp; &amp;</th>
<th>and</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>!</td>
<td>not</td>
</tr>
</tbody>
</table>
Conditionals (cont’d)

```java
switch ( ordinal_expression ) {
    case ordinal_value: {
        // ...
        break;
    }
    case ordinal_value: {
        // ...
        break;
    }
    default: {
        // ...
    }
}
```
Conditionals (cont’d)

```
switch ( ordinal_expression ) {
    case ordinal_value: {
        // ...
        break;
    }
    case ordinal_value: {
        // ...
        break;
    }
    default: {
        // ...
    }
}
```
Pointers
float x = 6.5;
float* px = &x;
#include <stdio.h>
define SIZE 10

int main() {
    float A[SIZE];
    int i;

    for ( i=0; i<SIZE; i++ )
        A[i] = i;

    for ( i=0; i<SIZE; i++ )
        printf( "A[%d] = %1.2f\n", i, A[i]);
}
#include <stdio.h>
#define SIZE 10

void main() {
  float A[SIZE];
  float* p;
  int i;

  p = A;
  for ( i=0; i<SIZE; i++ ) {
    *p = i;
    p++;
  }

  p = A;
  for ( i=0; i<SIZE; i++ ) {
    printf( "p=%p A[%d] = %1.2f *p = %1.2f\n",
            p, i, A[i], *p );
    p = p + 1;
  }
}
a.out
p=0x7fff88d54560 A[0] = 0.00 *p = 0.00
p=0x7fff88d54564 A[1] = 1.00 *p = 1.00
p=0x7fff88d54568 A[2] = 2.00 *p = 2.00
p=0x7fff88d5456c A[3] = 3.00 *p = 3.00
p=0x7fff88d54570 A[4] = 4.00 *p = 4.00
p=0x7fff88d54574 A[5] = 5.00 *p = 5.00
p=0x7fff88d54578 A[6] = 6.00 *p = 6.00
p=0x7fff88d5457c A[7] = 7.00 *p = 7.00
p=0x7fff88d54580 A[8] = 8.00 *p = 8.00
p=0x7fff88d54584 A[9] = 9.00 *p = 9.00
Arrays

```c
TYPE v[DIM];
TYPE* pv;

pv = v;
```
Arrays

- The name of an array is a pointer to the first cell of the array.

```c
char name[DIM];
```

- name is the same as `&(name[0])`
• * has two meanings, depending on context
  • “Pointer to”
  • “Contents of”

• & means “the address of”
int A[DIM];
int* p = A;   // “int pointer p”
int *q = &A[0]; // “int pointer q”

*p = 3;  // what p is pointing to gets 3
*(q+1) = 5; // what q is pointing to gets 5
#define DIM 10
int A[DIM];
int B[DIM];
int i;

for ( i=0; i<DIM; i++ ) A[i] = 13*i % 11;

• Write a C program that copies Array A into Array B using indexing, and then using pointers.
• Write a C program that finds the largest integer in Array A, using pointers.

```c
#define DIM 10
int A[DIM];
int i;

for ( i=0; i<DIM; i++ ) A[i] = 13*i % 11;
```
Homework-Related Exercise

Duffy Duck, (413) 585-2700, xxxxxxxx
Mickey Mouse, (617) 123-4567, yyyyyyyyyy
Minnie Mouse, (617) 123-4567, zzzzzz zzz
Bruno The Dog, (212) 678-9999, woof woof

• Given a list of names and personal information, blank out the phone numbers, leaving only the area code visible.
Functions

- Functions are always declared before they are used.
- Functions can return values of simple types (int, char, floats), and even pointers.
- Functions get parameters of simple types, and pointers.
- Passing by value is automatic. Passing by reference requires passing a pointer.
#include <stdio.h>

int sum( int a, int b ) {
    return a+b;
}

void main() {
    int x = 10;
    int y = 20;
    int z;

    z = sum(x, y);
    printf( "z = %d\n", z );

    z = sum(3, 8);
    printf( "z = %d\n", z );

    printf( "sum( 11, 22 ) = %d\n", sum(11, 22) );
}
#include <stdio.h>

void sum2( int a, int b, int  c ) {
    c = a+b;
}

void main() {
    int x = 10;
    int y = 20;
    int z;

    sum2( x, y,  z );
    printf( "z = %d\n", z );

    sum2( 3, 8,   x );
    printf( "x = %d\n", x );
}

z = 30
x = 11
#include <stdio.h>

void main() {
    int age;
    float myPi;
    char name[80];

    printf( "Enter your name, please: " );
    fgets( name, sizeof(name), stdin );
    // will truncate to first
    // 80 chars entered

    printf( "Enter your age: " );
    scanf( "%d", &age );

    printf( "Enter your version of pi: " );
    scanf( "%f", &myPi );

    printf( "%s is %d years old, and thinks pi is %1.10f
\n\n", name, age, myPi );
}
a.out
Enter your name, please: **Mickey**
Enter your age: **21**
Enter your version of pi: **3.14159**
Mickey is 21 years old, and thinks pi is 3.1415901184
Write a C program (no functions) that finds the smallest, the largest, and computes the sum of all the ints in A.

Write another program that does the same thing but uses functions. The results are passed back using return statements.
Exercise

- Write another program that does the same thing but uses functions, and this time the results are passed back via a parameter passed by reference.
We Stopped Here Last Time
Trip Through Memory Lane...
<table>
<thead>
<tr>
<th>Feature</th>
<th>Sharp MZ-1100</th>
<th>Sharp MZ-1200</th>
</tr>
</thead>
<tbody>
<tr>
<td>Floppy-disk drives</td>
<td>2 5½-inch</td>
<td>2 3½-inch</td>
</tr>
<tr>
<td></td>
<td>floppy-disk</td>
<td>floppy-disk</td>
</tr>
<tr>
<td></td>
<td>drives</td>
<td>drives</td>
</tr>
<tr>
<td>Display</td>
<td>LCD</td>
<td>LCD</td>
</tr>
<tr>
<td></td>
<td>40 characters</td>
<td>40 characters</td>
</tr>
<tr>
<td></td>
<td>by 8 lines</td>
<td>by 8 lines</td>
</tr>
<tr>
<td>Text processor</td>
<td>n.a.</td>
<td>n.a.</td>
</tr>
<tr>
<td>Text processor and</td>
<td>Vedit</td>
<td>Word Right</td>
</tr>
<tr>
<td>communications package</td>
<td></td>
<td>word-processing</td>
</tr>
<tr>
<td></td>
<td></td>
<td>package</td>
</tr>
<tr>
<td>Modem and card slot for use</td>
<td>built-in 300-bps</td>
<td>optional CRT display</td>
</tr>
<tr>
<td></td>
<td>modem and</td>
<td>display</td>
</tr>
<tr>
<td></td>
<td>card slot</td>
<td></td>
</tr>
<tr>
<td></td>
<td>for use with</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RCA CMOS</td>
<td></td>
</tr>
<tr>
<td></td>
<td>microboards</td>
<td></td>
</tr>
</tbody>
</table>

September 1983 © BYTE Publications Inc.
Photo 3: Space limitations on the front panel forced the designers of the Epson HX-20 to use a 20-character by 4-line display.

Photo 4: The Kaypro II has the ability to read and write to a variety of disk formats.

low power consumption. The recent improvement in chip-manufacturing technology has lowered the price and increased the performance of these memory devices. CMOS chips are still slower in operation than corresponding TTL chips.

height and microfloppy-disk drives, floppy disks remain the primary data-storage medium. The disadvantages of floppy disks include occasional incompatible recording formats for 5¼-inch disks and the con-

with its 10-megabyte hard disk and the Starlite HD20 (see photo 6) with its 20-megabyte hard disk can serve users with very large data-storage needs. Both have a staggering
Photo 5: With its 10-megabyte hard disk preloaded with the bundled software, the Kaypro-10 can store 50 disks’ worth of information.

Photo 6: The Starlite HD 20 offers a staggering 20 megabyte storage in a portable computer.

But don’t let the lure of bundled software sway your decision on which portable computer to buy. You may not like a particular software package that is included with the portable computer you choose. Selecting software is sometimes the most important aspect of choosing a portable computer. The software you choose should be compatible with other software packages you hope to use.

Processor, the de facto operating system is CP/M. Compatibility is reasonably assured regardless of the type of computer, magnetic media, or display used. Practically every major software application package is available in a CP/M format. By using a

you can create a data file with an electronic spreadsheet program on the IBM and use that data on a compatible portable as long as you have the same version of the spreadsheet program for the portable.

The last level of compatibility in
Photo 1: Visicalc spreadsheets can be viewed on the HP-75's single-line display, on a video monitor, or on printouts.
The Gavilan—Full-Function Portable Computer

by F. John Zepecki

the traveling professional. The computer had to be able to run for a long time on an internal battery pack (ideally, for at least 8 hours without recharging), yet it could not weigh more than 15 pounds. It had to have a standard QWERTY keyboard with a numeric 10-key pad (we felt that

tal halves of 32 rows each, and the drivers are multiplexed so that both halves are scanned simultaneously. As a result, the display controller supplies two data streams, one for each half of the display. This way, the display’s 64 rows are painted in the time it takes to paint 32 rows.

Although getting an 8-line, 80-character display in the limited space available was an achievement, an 8-line page restricted our ability to process lengthy files. To simplify this, a special Zoom function was added in firmware. The Zoom function presents an outline image of a document with the positions of the eight active lines shown in a rectangular overlay. Using this function, the overlay can be placed anywhere on the page outline, and the enclosed eight lines are displayed by shifting the display's horizontal offset.

The mechanical interface is made by hinges fastened to the top of the posts and bottom, or screen side, of the display lid (see photo 2). The electrical interface is provided by a ribbon cable in one of the posts. To operate the computer, the user pushes a button on the right side of the case, releasing the posts, which lift the bottom of the display lid above the top of the computer case. The lid then swings manually into an upright position. Mechanical detents providing 15-degree indexing beginning at 90 degrees let the user lock the display in place.
Photo 1a and 1b: The Gavilan mobile computer offers a full-size, full-travel keyboard with a numeric 10-key pad and a fold-up 8-line, 80-character liquid-crystal display. The snap-on printer adds less than 5 inches to the computer’s length, yet provides 50 cps throughput onto standard plain paper. The computer and printer fit easily into a standard-size briefcase and weigh less than 15 pounds.
<table>
<thead>
<tr>
<th>Company</th>
<th>Model/Description</th>
<th>Dimensions</th>
<th>Weight</th>
<th>Power</th>
<th>Model Type</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Osborne Computer Corp.</td>
<td>Executive II</td>
<td>20½ by 13 by 9</td>
<td>28 lbs.</td>
<td>AC</td>
<td>$3195</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Osborne 1</td>
<td>20½ by 14½ by 8½</td>
<td>23 lbs., 8 oz.</td>
<td>AC</td>
<td>$1795</td>
<td></td>
</tr>
<tr>
<td>Otrona Corp.</td>
<td>Attache</td>
<td>12 by 13½ by 5¾</td>
<td>18 lbs.</td>
<td>120 or 220V AC</td>
<td>$3995</td>
<td></td>
</tr>
<tr>
<td>Panasonic Co. Secaucus, NJ</td>
<td>Hand-Held Computer RL-H1800</td>
<td>1 by 9 by 3</td>
<td>21.9 oz.</td>
<td>batteries or AC</td>
<td>$380</td>
<td></td>
</tr>
<tr>
<td>Panasonic Co. Secaucus, NJ</td>
<td>JR-800</td>
<td>10¼ by 5½ by 1¾</td>
<td>1 lb., 10 oz.</td>
<td>batteries or AC</td>
<td>$499.95</td>
<td></td>
</tr>
<tr>
<td>Company</td>
<td>Model</td>
<td>Size</td>
<td>Weight</td>
<td>Power Supply</td>
<td>Price</td>
<td>Chipset</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>----------------------</td>
<td>------------</td>
<td>--------</td>
<td>-----------------------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>Non-Linear Systems Solana Beach, CA</td>
<td>Kaypro 10</td>
<td>19 by 16 by 8</td>
<td>27 lbs</td>
<td>110 or 220V AC</td>
<td>$2795</td>
<td>Z80A</td>
</tr>
<tr>
<td>Olympia USA Inc. Somerville, NJ</td>
<td>Portable Computer OL-H004</td>
<td>1 1/4 by 9 by 3 3/4</td>
<td>21 oz</td>
<td>batteries or AC</td>
<td>$380</td>
<td>proprietary</td>
</tr>
<tr>
<td>Olympia USA Inc. Somerville, NJ</td>
<td>Portable Computer OL-0008</td>
<td>1 1/2 by 9 by 3 3/4</td>
<td>21 oz</td>
<td>batteries or AC</td>
<td>$480</td>
<td>proprietary</td>
</tr>
<tr>
<td>Osborne Computer Corp. Hayward, CA</td>
<td>The Executive</td>
<td>20 1/2 by 13 by 9</td>
<td>28 lbs</td>
<td>AC</td>
<td>$2495</td>
<td>Z80A</td>
</tr>
<tr>
<td>Osborne Computer Corp. Hayward, CA</td>
<td>Executive II</td>
<td>20 1/2 by 13 by 9</td>
<td>28 lbs</td>
<td>AC</td>
<td>$3195</td>
<td>8088</td>
</tr>
<tr>
<td>Osborne Computer Corp. Hayward, CA</td>
<td>Osborne 1</td>
<td>20 1/2 by 14 1/2 by 8 1/2</td>
<td>23 lbs., 8 oz.</td>
<td>AC</td>
<td>$1795</td>
<td>Z80A</td>
</tr>
<tr>
<td>Otrona Corp. Boulder, CO</td>
<td>Attache</td>
<td>12 by 13 1/2 by 5 3/4</td>
<td>18 lbs</td>
<td>120 or 220V AC</td>
<td>$3995</td>
<td>Z80A</td>
</tr>
<tr>
<td>Panasonic Co. Secaucus, NJ</td>
<td>Hand- Held Computer RL-H1800</td>
<td>1 by 9 by 3</td>
<td>21.9 oz</td>
<td>batteries or AC</td>
<td>$380</td>
<td>proprietary</td>
</tr>
<tr>
<td>Panasonic Co. Secaucus, NJ</td>
<td>JR-800</td>
<td>10 1/4 by 5 1/2 by 1 1/2</td>
<td>1 lb., 10 oz.</td>
<td>batteries or AC</td>
<td>$499.95</td>
<td>80C85</td>
</tr>
</tbody>
</table>
Dynamic Variables
• Dynamic: think "new" in Java

• Memore Allocation for New Data Structure = malloc()
#include <stdio.h>
#include <stdlib.h>

void main() {
    int *A, N, i, smallest;

    printf( "How many ints? " );
    scanf( "%d", &N );
    A = (int *) malloc( N * sizeof( int ) );

    for ( i=0; i<N; i++ ) {
        printf( "> " );
        scanf( "%d", &A[i] );
    }

    smallest = A[0];
    for ( i=1; i<N; i++ )
        if ( A[i] < smallest )
            smallest = A[i];

    free( A );

    printf( "The smallest = %d\n", smallest );
}
Exercise

- Modify the previous program that computed the min and max of an array, but this time you allocate the array dynamically from the user input, i.e. ask the user for number of ints, then the ints.

  (use `scanf("%d",&x)` to get an int from keyboard into x)
sizeof() can be tricky...
```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main( int argc, char* argv[] ) {
    int A[] = { 1, 2, 3, 4, 5 };
    int *B = (int *) malloc( 5 * sizeof( int ) );
    int *p = A;
    char name[] = "Smith College";
    int a = 3;
    float x = 3.14159;
    int i;

    for ( i=0; i<5; i++ ) B[i] = i;

    printf( "sizeof(A)    = %lu\n", sizeof( A ) );
    printf( "sizeof(A[0]) = %lu\n", sizeof( A[0] ) );
    printf( "sizeof(B)    = %lu\n", sizeof( B ) );
    printf( "sizeof(B[0]) = %lu\n", sizeof( B[0] ) );
    printf( "sizeof(p)    = %lu\n", sizeof( p ) );
    printf( "sizeof(*p)   = %lu\n", sizeof( *p ) );
    printf( "sizeof(name) = %lu\n", sizeof( name ) );
    printf( "strlen(name) = %lu\n", strlen( name ) );
    printf( "sizeof(a)    = %lu\n", sizeof( a ) );
    printf( "sizeof(x)    = %lu\n", sizeof( x ) );

    return 0;
}
```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main( int argc, char* argv[] ) {
    int A[] = { 1, 2, 3, 4, 5 };
    int *B = (int *) malloc( 5 * sizeof( int ) );
    int *p = A;
    char name[] = "Smith College";
    int a = 3;
    float x = 3.14159;
    int i;

    for ( i=0; i<5; i++ ) B[i] = i;

    printf( "sizeof(A)    = %lu\n", sizeof( A ) );
    printf( "sizeof(A[0]) = %lu\n", sizeof( A[0] ) );
    printf( "sizeof(B)    = %lu\n", sizeof( B ) );
    printf( "sizeof(B[0]) = %lu\n", sizeof( B[0] ) );
    printf( "sizeof(p)    = %lu\n", sizeof( p ) );
    printf( "sizeof(*p)   = %lu\n", sizeof( *p ) );
    printf( "sizeof(name) = %lu\n", sizeof( name ) );
    printf( "strlen(name) = %lu\n", strlen( name ) );
    printf( "sizeof(a)    = %lu\n", sizeof( a ) );
    printf( "sizeof(x)    = %lu\n", sizeof( x ) );

    return 0;
}
```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main( int argc, char* argv[] ) {
    int A[] = { 1, 2, 3, 4, 5 };
    int *B = (int *) malloc( 5 * sizeof( int ) );
    int *p = A;
    char name[] = "Smith College";
    int a = 3;
    float x = 3.14159;
    int i;

    for ( i=0; i<5; i++ ) B[i] = i;

    printf( "sizeof(A)    = %lu\n", sizeof( A ) );
    printf( "sizeof(A[0]) = %lu\n", sizeof( A[0] ) );
    printf( "sizeof(B)    = %lu\n", sizeof( B ) );
    printf( "sizeof(B[0]) = %lu\n", sizeof( B[0] ) );
    printf( "sizeof(p)    = %lu\n", sizeof( p ) );
    printf( "sizeof(*p)   = %lu\n", sizeof( *p ) );
    printf( "sizeof(name) = %lu\n", sizeof( name ) );
    printf( "strlen(name) = %lu\n", strlen( name ) );
    printf( "sizeof(a)    = %lu\n", sizeof( a ) );
    printf( "sizeof(x)    = %lu\n", sizeof( x ) );

    return 0;
}
```
#include <stdio.h>

void main() {
    FILE *fp;
    int i;
    char name[] = "Smith College";

    fp = fopen("hello.txt", "w"); // open file for writing

    fprintf(fp, "\nHello "); // write constant string
    fprintf(fp, "%s\n\n", name ); // write string

    fclose(fp); // close file
}
```c
#include <stdio.h>

void main() {
    FILE *fp;
    char line[80];

    fp = fopen( "fgets2.c", "r" );  // open file for reading

    while ( !feof( fp ) ) {       // while not eof
        fgets( line, 80, fp );     // get at most 80 chars
        if ( feof( fp ) )          // if eof reached stop
            break;
        line[79] = '\0';           // truncate line to be safe
        printf( "%s", line );     // print it
    }

    fclose( fp );                // close file
}
```
File I/O: Input Numbers

[~handout] `cat fileOfInts.txt`

```
4
1 2 3
4 5 6
7 8 9
10 11 12
```
#include <stdio.h>

void main() {
    FILE *fp;
    char line[80];
    int N, n1, n2, n3;

    fp = fopen( "fileOfInts.txt", "r" ); // 1st number is # of lines
    // then 3 ints per line

    if ( feof( fp ) ) {
        printf( "Empty file!\n\n" );
        return;
    }

    // get the number of lines
    fscanf( fp, "%d", &N );

    while ( !feof( fp ) ) {
        fscanf( fp, "%d %d %d", &n1, &n2, &n3 );
        if ( feof( fp )
            break;
        printf( "%d, %d, %d\n", n1, n2, n3 );
    }

    fclose( fp );
}
File I/O: Reading Ints

[~handout] a.out
1, 2, 3
4, 5, 6
7, 8, 9
10, 11, 12
Exercise

• At the Bash prompt type:

```bash
cat > data.txt
4
10
1
2
3
^D
```

• Write a program that reads the file data.txt, which has the number of ints in contains on the first line, then one int per line for the rest of the file. Your program must use a dynamically created array to store the numbers, and find the min and max of the array, and print them.
Function Prototypes and Multiple-File Projects
```c
#include <stdio.h>
#include <stdlib.h>

#define N 10

int smallest( int* A ) {
    int i, min = A[0];
    for (i=0; i<N; i++ )
        if (A[i]<min ) min=A[i];
    return min;
}

void largest( int A[], int *max ) {
    int i;
    *max = A[0];
    for (i=0; i<N; i++ )
        if (A[i]>*max ) *max=A[i];
}

void addition( int A[], int *x ) {
    int i, sum=0;
    for (i=0; i<N; i++ )
        sum +=A[i];
    *x = sum;
}

// functions go here...
void main() {
    int A[N] = { 3, 2, 1, 0, 6, 5, 9,8, 7 };
    int min, max, sum;

    min = smallest( A );
    largest( A, &max );
    addition( A, &sum );

    printf( "%d %d %d\n", min, max, sum );
}
```
myfuncts.h

#ifndef MYFUNCS_H
#define MYFUNCS_H

//—— prototypes ——
int smallest( int* A, int N );
void largest( int A[], int N, int *max );
void addition( int A[], int N, int *x );

#endif

myfuncts.c

#include "myFuncs.h"

int smallest( int* A, int N ) {
    int i, min = A[0];
    for (i=0; i<N; i++)
        if (A[i]<min ) min=A[i];
    return min;
}

void largest( int A[], int N, int *max ) {
    int i;
    *max = A[0];
    for (i=0; i<N; i++)
        if (A[i]>*max ) *max=A[i];
}

void addition( int A[], int N, int *x ) {
    int i, sum=0;
    for (i=0; i<N; i++)
        sum +=A[i];
    *x = sum;
}

smallestLargestSum3.c

#include <stdio.h>
#include <stdlib.h>
#include "myFuncs.h"
#define N 10

void main() {
    int A[N] = { 3, 2, 1, 0, 6, 5, 9, 8, 7 };
    int min, max, sum;

    min = smallest( A, N );
largest( A, N, &max );
addition( A, N, &sum );

    printf( "%d %d %d\n", min, max, sum );
}
myfuncs.h

#ifndef MYFUNCS_H
#define MYFUNCS_H

//—— prototypes ——
int smallest( int* A, int N );
void largest( int A[], int N, int *max );
void addition( int A[], int N, int *x );

#endif

myfuncs.c

#include "myFuncs.h"

int smallest( int* A, int N ) {
    int i, min = A[0];
    for (i=0; i<N; i++)
        if (A[i]<min ) min=A[i];
    return min;
}

void largest( int A[], int N, int *max ) {
    int i;
    *max = A[0];
    for (i=0; i<N; i++)
        if (A[i]*max ) *max=A[i];
}

void addition( int A[], int N, int *x ) {
    int i, sum=0;
    for (i=0; i<N; i++)
        sum +=A[i];
    *x = sum;
}