

Phys4051: C Lecture 2 & 3

Functions (Review)
Comment Statements
Variables & Operators
Branching Instructions

Comment Statements

■ Method 1: `/* */` ■ Method 2: `//`

```
/* Single Line */       //Single Line
```

```
/* This comment       // This comment  
  spans more than one // spans more than  
  line.                // spans more than  
*/                     // one line.
```

```
                       int x; //variable x
```

2

C Data Types

- No Return Value:
(For functions & pointers only!)
 - `void`
- Integers:
 - `char`
 - `short`
 - `int`
 - `long`
- Real (floating point):
 - `float`
 - `double`

3

Data Types: Signed / Unsigned

- All integer data type declarations can be signed (default) or unsigned.
- Example:
 - `unsigned int ival = 134;`
 - `signed int sval = -107;`

4

Global / Local Variables

- All variables declared outside of a function are GLOBAL.
- All variables declared within a function are LOCAL to that function.
- Global variables have global scope, i.e., they are seen in ALL the functions.
- Local variables have local scope. They "exist" only within the function that declares them.

5

Variable Declarations

- Position in Program:

Global vs. Local

Always declare variables before any statements or before main!

6

Variable Initialization

- Global Variables:
 - Always initialized to 0
- Local Variables:
 - Never initialized
 - Random

7

Variable Declaration & Initialization

- Example 1a:
 - short x;
 - short y;
 - short z;

 - x = 3;
 - y = 4;
- Example 1b:
 - short x = 3,
 - y = 3, z;

8

C-Operators

- Mathematical
- Relational
- Assignment
 - Logical
 - Bitwise



C-Operators

- Are always executed:
 - 1: Left to right
 - 2: By order of precedent

10



Mathematical C-Operators

- By order of precedence:
 - * Multiplication
 - / Division
 - % Remainder: ex $13\%4 = 1$
 - + Addition
 - - Subtraction

11



Mathematical C-Operators:

- Accuracy:
 - When applying an operator to variables having different levels of accuracy, then the calculation is carried out using the level of accuracy as specified by the variable with the higher level of accuracy.
- Example:

12

Mathematical C-Operators: Accuracy Example

```
short k, m = 2, n = 3;
float a, b = 2.0, c = 3.0;
```

```
k = m / n;
k = b / n;
```

```
a = m / n;
a = 1.0 * m / n;
a = (float) m / n; //type casting
a = b / c;
```

```
k = ?
k = ?
```

```
a = ?
a = ?
a = ?
a = ?
```

13

Mathematical C-Operators: Shortcut Notations

Equivalent Notations:

```
x = x + 1;
x = x - 1;
```

```
x++; ++x;
x--; --x;
```



```
a = a + 3.14;
a = a - 3.14;
a = a * 3.14;
a = a / 3.14;
```

```
a += 3.14;
a -= 3.14;
a *= 3.14;
a /= 3.14;
```

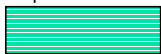
14

C-Operators: Shortcut Notations Example

Example 1:

```
int y, x = 4051;
y = ++x;
printf("%d", y);
```

Output:



Example 2:

```
int y, x = 4051;
y = x++;
printf("%d", y);
```

Output:



15

Branching Instructions: Simple if-statement

■ Example1:

```
■ if ( logic condition )  
{  
    statement(s);  
}
```

```
■ if ( x > 1 )  
{  
    x = 0;  
    printf("Hello");  
}
```

16

Logic Condition

■ TRUE:

Anything, except 0 or
FALSE

■ FALSE:

Only 0 or FALSE

■ Example:

```
if( 3 )  
{  
    printf("always true");  
}
```

■ Example:

```
if( 0 )  
{  
    printf("never true");  
}
```

17

if-else Statement

```
■ if( logic condition )  
{  
    statement(s); //executed only if logic  
                //condition is "TRUE"  
}  
else  
{  
    statement(s); //executed only if logic  
                //condition is "FALSE"  
}
```

18

if-statement Shortcut Notation

(1)

- If the if-statement contains only ONE statement, braces can be omitted:

```
if( x )
{
    printf("TRUE");
}
else
{
    printf("FALSE")
}
```

Equivalent program segment:

```
if( x )
    printf("TRUE");
else
    printf("FALSE");
```

19

if-statement Shortcut Notation

(1) **WARNING!!!**

```
if( x )
{
    printf("TRUE");
}
else
{
    printf("FALSE");
    printf("AGAIN");
}
printf("DONE")
```

```
if( x )
    printf("TRUE");
else
    printf("FALSE");
    printf("AGAIN");
printf("DONE");
```

20

Relational Operators (1)

- < Less Than
- <= Less Than and Equal
- > Greater Than
- >= Greater Than and Equal

- == Equal**
- != Not Equal**

21

Relational Operators (2)

WARNING: Equal Operator

Example 1:

```
short x = 5;
if( x = 3 )
{
    printf("x is 3");
}
```

Output:



Example 2:

```
short x = 5;
if( x == 3 )
{
    printf("x is 3");
}
```

Output:



22

Relational Operators (3)

The "Not" (!) Operator

What is the difference between the following two program segments?

```
if( !x )
{
```

```
if( x == 0 )
{
```

23

Logic Operators: AND & OR

(1)

The following operators act on the entire variable or expression as a whole and return either TRUE (1) or FALSE (0)

&& AND

|| OR (shift "\ " key)

24

Logic Operators: (Bytewise) AND & OR (2)

- The AND and OR operators are often used to link relational operators.
- Example:
How do you write `3 < x < 10` in C?

25

Logic Operators: Bitwise Operators (1)

<< Left Shift
>> Right Shift
~ Bitwise NOT
& Bitwise AND
^ Bitwise XOR
| Bitwise OR

- Bitwise operators operate on a variable or expression "bit by bit."

26


"Bytewise" and Bitwise Operators (2)

- Example 1:
"Bytewise"
`short a = 5, b = 10, y;`
`y = a && b;`

`y = ???`
- Example 2: Bitwise
`short a = 5, b = 10, y;`
`y = a & b;`

`y = ???`

27



Logic Operators: Bitwise Operators (3)

- Bitwise operators are often used in hardware control applications
- Example 1:
Check one specific bit in a byte
(BIT MASKING)
- Example 2:
Toggle a specific bit in a byte

28

Typical C-Function Syntax

Function Declaration `double Thdeg(float x, float y) ;`

Function Header `double Thdeg(float x, float y)`

Local Variable Declarations `{
float fthetadeg, fthetarad ;
float fPi = 3.1415 ;
double ftan ;`

Statements and Function Calls `ftan = y / x ;
fthetarad = atan (ftan) ;
fthetadeg = ftheta * 180.0/fPi ;

return (fthetadeg) ;
}`

Data Type Range

Bytes	Signed Declaration	Range	Unsigned Declaration	Range
1	char	-128 to 127	unsigned char	0 to 255
2	short (or: int)	-32768 to 32767	unsigned short (unsigned int)	0 to 65635
4	long	-2147483648 to 2147483647		0 to 4294967295
4	float	$\pm 3.4 \text{ E} \pm 38$		
8	double	$\pm 1.7 \text{ E} \pm 308$		

Program Global & Local Variables

```
#include <ansi_c.h>

void F1( short x);
void F2( short x);

short s1;

main()
{
    short sm1;
    short sm2 = 123;

    //Non-intialized values
    printf("Part 1: Non-Initialized Variables\n");
    printf("s1: %d\n", s1);
    printf("sm1: %d\n\n", sm1);

    printf("Part 2: Passing a Variable to a Function\n");
    printf("sm2: %d\n", sm2);
    F1( sm2 ); //passing a variable by value
    printf("sm2: %d\n\n", sm2);

    printf("Global vs. Local Variables\n");
    F2( sm2 ); //global vs. local variables
    printf("sm2: %d\n", sm2);
    printf("s1: %d\n\n", s1);

}

void F1( short x)
{
    printf("before: x: %d\t", x);
    x = 888;
    printf("after: x: %d\n", x);
}

void F2( short x)
{
    short sm2 = -111;
    s1 = -1234;
}
```

Output:

Part 1: Non-Initialized Variables

s1:

sm1:

Part 2: Passing a Variable to a Function

sm2:

before: x: after: x:

sm2:

Global vs. Local Variables

sm2:

s1:

1) ANSI-C / No GUI

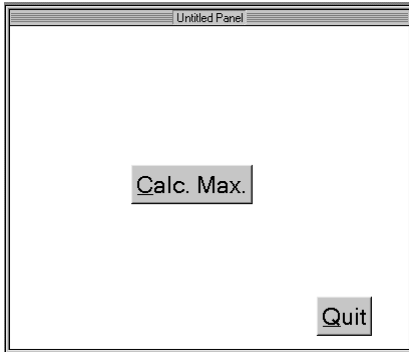
```
//Program MyMaxNoGui
#include <ansi_c.h>
float MyMax(float x, float y);

main()
{
    float u = 123.0, v = -543.2;
    float res;

    res = MyMax( u, v);
    printf("Maximum is: %f\n", res);
}

float MyMax(float x, float y)
{
    float max;
    if( x > y)
    {
        max = x;
    }
    else
    {
        max = y;
    }
    return max;
}
```

2) LabWindows / With GUI



2a) LabWindows Generated Code:

```
#include <cvirte.h>
#include <userint.h>
#include "MyMaxGui.h"

static int panelHandle;

int main (int argc, char *argv[])
{
    if (InitCVIRTE (0, argv, 0) == 0)
        return -1; /* out of memory */
    if ((panelHandle = LoadPanel (0, "MyMaxGui.uir", PANEL)) < 0)
        return -1;
    DisplayPanel (panelHandle);
    RunUserInterface ();
    return 0;
}

int CVICALLBACK Quit (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{
    switch (event) {
        case EVENT_COMMIT:
            QuitUserInterface (0);
            break;
    }
    return 0;
}

int CVICALLBACK CalcMax (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{
    switch (event) {
        case EVENT_COMMIT:

            break;
    }
    return 0;
}
```

2b) Code for MyMax function added (in Bold):

```
#include <ansi_c.h>
#include <cvirte.h>
#include <userint.h>
#include "MyMaxGui.h"
static int panelHandle;
float MyMax(float x, float y);

int main (int argc, char *argv[])
{
    if (InitCVIRTE (0, argv, 0) == 0)
        return -1; /* out of memory */
    if ((panelHandle = LoadPanel (0, "MyMaxGui.uir", PANEL)) < 0)
        return -1;
    DisplayPanel (panelHandle);
    RunUserInterface ();
    return 0;
}

int CVICALLBACK Quit (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{
    switch (event) {
        case EVENT_COMMIT:
            QuitUserInterface (0);
            break;
    }
    return 0;
}

int CVICALLBACK CalcMax (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{
    float u = 123.0, v = -543.2;
    float res;

    switch (event) {
        case EVENT_COMMIT:

            res = MyMax( u, v);
            printf("Maximum is: %f\n", res);

            break;
    }
    return 0;
}
```

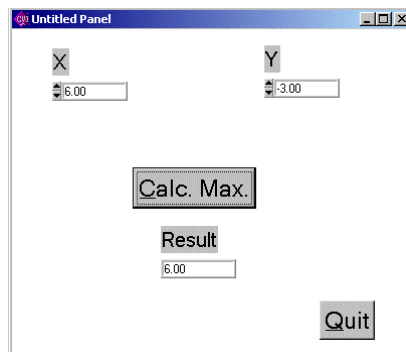
```

float MyMax(float x, float y)
{
    float max;
    if( x > y)
        {
            max = x;
        }
    else
        {
            max = y;
        }
    return max;
}

```

2c) Code Added to Read Input and to Display the Result:

(Changes from previous version are shown in bold)



```

#include <ansi_c.h>
#include <cvirte.h> /* Needed if linking in external
compiler; harmless otherwise */
#include <userint.h>
#include "MyMaxGui.h"

static int panelHandle;
float MyMax(float x, float y);

int main (int argc, char *argv[])
{
    if (InitCVIRTE (0, argv, 0) == 0) /* Needed if linking in
external compiler; harmless otherwise */
        return -1; /* out of memory */
    if ((panelHandle = LoadPanel (0, "MyMaxGui.uir", PANEL)) < 0)
        return -1;
    DisplayPanel (panelHandle);
    RunUserInterface ();
}

```

```

    return 0;
}

int CVICALLBACK Quit (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{
    switch (event) {
        case EVENT_COMMIT:
            QuitUserInterface (0);
            break;
    }
    return 0;
}

int CVICALLBACK CalcMax (int panel, int control, int event,
    void *callbackData, int eventData1, int eventData2)
{
    float u, v;
    float res;

    switch (event) {
        case EVENT_COMMIT:
            GetCtrlVal (panelHandle, PANEL_XINPUT, &u);
            GetCtrlVal (panelHandle, PANEL_XINPUT, &v);

            res = MyMax( u, v);

            SetCtrlVal (panelHandle, PANEL_RESULT, res);
            break;
    }
    return 0;
}

float MyMax(float x, float y)
{
    float max;

    if( x > y)
    {
        max = x;
    }
    else
    {
        max = y;
    }
    return max;
}

```