

# Computer Science & Engineering 150A

## Problem Solving Using Computers – Laboratory

### Lecture 15 - Reviews

Shuai Xie

(Adapted from Derrick Stolee, Lin Liu & Shuai Xie )

Spring 2010

- no: \*, #, &, %, ...
- no: space
- cannot start with numbers
- no: reserved words, e.g. int double char
- ...
- Bo\*ys, #a, angle's, char

Var Name	Stores	Scanf	Printf
int	Integers	%d	%d
double	Decimals	%lf	%f
char	Letters	%c	%c

- `char grade = 'A';`
- `%10d`      `%7.2f`
- Type cast
  - `int i=3, j=4;`
  - `int k; k=i/j; (k=0)`
  - `double l;`
  - `l=(double)i/j (l=0.75);`

# Some Operators

CSCE150A

- Math operators:  $+$   $-$   $*$   $/$   $\%(\text{remainder})$
- Relation and equality operators:  $<$   $>$   $<=$   $>=$   
 $==$   $!=$
- logical operators:  $\&\&$   $\|\$   $!$
- Notes: order of operators; using " $( )$ ".

- Math function: `abs(x)`, `fabs(x)`, `sin(x)`, `cos(x)`, `pow(x,y)`, `sqrt(x)`
- `#include <math.h>`
- Math expression:

$$\left| \frac{1}{\sqrt{1-x^2}} \right| \Rightarrow \text{fabs}(1.0/\text{sqrt}(1-\text{pow}(x,2)))$$

```
int i=1, total=0;

while ( i <=10 )
{
    total = total + i;
    i++;
}
```

# do while loop

CSCE150A

- Repetition: At least once

```
do
{
    executable1;
    ...
    change variable in conditional statement;
} while ( check condition );
```

- Similar to while loop
- Execute loop body at least once
- Don't miss ";"

# do while loop

CSCE150A

```
int i=1, total=0;

do
{
    total = total + i;
    i++;
} while ( i >= 10 );
```



```
for ( i = 1; i < 10; i++ )  
{  
    total = total + i;  
}
```

# Switch statement

CSCE150A

```
switch ( controlling_variable )
{
    case 1:
        executable group 1 ;
        break ;
    case 2:
        executable group 2 ;
        break ;
    ...

    default:
        executable group last ;
}
```

- controlling\_variable: int, char
- char variable: 'c' ;
- break;
- other cases -- default :

```
int windlevel ;  
...  
switch ( windlevel )  
{  
    case 2:  
        printf("Level 2:  Breeze") ;  
        break ;  
    case 7:  
        printf("Level 7:  Strong wind") ;  
    case 10:  
        printf("Level 10:  Storm wind") ;  
        break ;  
  
    default:  
        printf("More detail see our website") ;  
}
```

# Function: Prototype

CSCE150A

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

```
void PrintAlarm(void);
```

```
int Cube(int x);
```

```
double EllipseArea(int x, int y);
```

```
int main(void)
```

```
{ ...
```

- Between preprocessor directive and main function
- Return type – void, int, double, char, etc
- Argument – void, single, multiple
- ";"

# Function: Definition

CSCE150A

```
ReturnType FuncName(type Arg 1,...)
{
    local variable declarations;
    executable statements;
    return statement;
}
```

- After the main function
- $\approx$  prototype, without ";"
- "{", "}" – scope
- Local variables disappear outside scope
- Output
  - Return statement
  - Match returntype

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

```
double CountArea(double x, double y); \\Prototype
```

```
int main(void)
```

```
{
```

```
...
```

```
area = CountArea(x, y); \\Call the function
```

```
...
```

```
}
```

```
double CountArea(double x, double y) \\Definition
```

```
{
```

```
return (x * y);
```

```
}
```

# Function with Output Parameters

CSCE150A

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

```
void CountPro(int x, int y, int *zp); \\Prototype
```

```
int main(void)
{
    int a=4,b=5,c=6;
    CountPro(c,a,&b); \\Call the function
    printf("The sum is %d",c);
    return 0;
}
```

```
void CountPro(int x, int y, int *zp) \\Definition
{
    *zp = x * y;
}
```

# Declaring and Referencing Arrays

CSCE150A

- An array is a single variable that references several data elements in a row.
- To define, declare the *name of the array* and the *number of cells* associated with it.

**Type ArrayName[Size];**

- Example:      `double x[8];`

Creates 8 memory cells of type `double` with the name `x`;

These memory cells will be adjacent to each other in memory.



- ...

```
#define SIZE 10  
  
...  
int array[SIZE];  
for(i=0; i < SIZE; i++)  
{  
    scanf("%d", &array[i]);  
}
```

- ...

```
for(i=0; i < SIZE; i++)  
{  
    printf("%d", array[i]);  
}
```

- A list of characters is a *string*.
- We use arrays of type `char`:

```
char str[] = "My string!";
```

'M'	'y'	' '	's'	't'	'r'	'i'	'n'	'g'	'!'	'\0'
77	121	32	115	116	114	105	110	103	33	0

- The final `'\0'` character signifies the end of the string.
- *Note: there may be more entries in the array, but they will be ignored!*
- `int size = strlen(str);` (size=10, and include `<string.h>`)

# Dealing with Character Values

CSCE150A

- Upper-case letters 'A' through 'Z' appear as contiguous numbers, in order.
- Lower-case letters 'a' through 'z' appear as contiguous numbers, in order.
- You can treat a char type (including string index) and char literal (such as 'a') as numbers!
  - You can use comparisons: == != <= >=
  - You can use operations: + - / \*

# Storing Strings

CSCE150A

- You can use static arrays to store strings.

```
char string1[200];
char string2[100] = "Length determined by 100";
```

- Then, you can output strings with printf:  

```
scanf("%s",string1);
printf("My string2 is:  %s\n",string2);
```