CSC 215
Types, Operators, and Expressions

Dr. Achraf El Allali

Contant

- Constants do not change during program execution
- Must be declared before use:

```
#define VTAB ‘\013’    /* ASCII vertical tab*/
#define RIS 0xD4
#define BO 37
#define PI 3.1415
#define NL “\n‟         /*new line*/
```

- #define is a preprocessor directive

Constants

```
#include<stdio.h>
#define PI 3.14    /* PI is a constant */
main ()
{
   int r;
   float area;
   scanf (“%d”, &r);
   area = PI * r * r;
   printf (“Area = %d”, area);
}
```

Arithmetic operators

- Binary operators: addition (+), subtraction (-), multiplication (*), division (/), remainder (%)

- Unary operators: unary plus (+), unary minus (-)

Swapping two numbers

- How do we swap two numbers x and y?

Swapping two numbers

```
int temp;
temp = x;
x = y;
y = temp;
```

Puzzle

- How do you swap two numbers without using

a temp variable?

  - Hint: Use arithmetic operators + and -

Relational Operators

- > , >=, <, <=, ==, !=
- Example

```
if (a < b)
    printf (“a is less than b\n”)
else
    printf (“b is less than or equal to a\n”);
```

Relational Operators

The result of comparison of two expressions is true if the condition is satisfied and false otherwise

• There is no special logical data type in C. The value of a relational expression is of type int:

– 15 > 10 has the value 1 (true)

– 15 < 10 has the value 0 (false)

Logical Operators

- &&, ||
- Example

```
if (a == 5 || b == 5 )
    printf (“There is at least one 5\n”)
else
    printf (“There isn't any 5\n”)
```

Logical Operators

- The operands may be of any arithmetic type while the result is always int

- The value of a logical expression is either 1 (true) or 0 (false)

Increment and Decrement Operators

- ++ , --

- Prefix increment

  c = 5;

  x = ++c; /* value of c is 6 and x is 6 */

Increment and Decrement Operators

- Postfix increment

```
c = 5;
x = c++;  /* value of c is 6 but x is 5 */
```

- Equivalent to:

```
x = c;
c = c + 1;
```

Bitwise Operators

- Bitwise AND &
- Bitwise OR |
- Bitwise exclusive OR ^
- Left shift <<
- Right shift >>
- One's complement ~

Bitwise Operators

- Examples:

  0110 & 0011 -> 0010

  0110 | 0011 -> 0111

  0110 ^ 0011 -> 0101

  01101110 << 2 -> 10111000

  01101110 >> 3 -> 00001101

  ~0011 -> 1100

  Notice: << and >> multiply/divide by 2n

Puzzle

- How do you find if a number is a power of 2?

```
if (…)
    printf (“Power of 2\n”);
else
    printf (“Not a power of 2\n”);
```

Hint: Use bitwise operator &

Assignment Operator

- An assignment expression is of the form:

variable= expression;

- The precedence of the assignment operator (=) is lower than that of the arithmetic operators:

  sum = sum + item;

  sum = (sum + item);

Assignment Operator

```
i = i + 2;
```

is equivalent to

```
i += 2;
```

```
i = i * 2;
```

is equivalent to

```
i *= 2;
```

Conditional Operator

- expression_1 ? expression_2 : expression_3
- Example, the maximum of two values:
  max = x > y ? x : y;

Conditional Expressions

```
if (a > b)
        z = a;
else
        z = b;
```

is equivalent to

```
z = (a > b) ? a : b;
```

Arrays

- Declaration:
  - int score[10];
- Reference:
  - score[3] = 10;
- Index starts from 0 (goes till 9)
- Can use variables for indexing
  - i = 5;
  - score[i] = 2;

Multi-dimensional Arrays

- Array of arrays
  - int a[3][3];
  - a[0][0] = 3;
  - a[1][2]=0

0	10
1	5
2	2

	0	1	2
0	3	10	2
1	9	5	0
2	5	6	1

Array Initialization

- int score[5] = { 41, 97, 10, 55, 100 };
- int a [5] = { 3, 2, 5 }; ≡ int a[5] = { 3, 2, 5, 0, 0 };
- int b[] = { 7, 11, 100, 4 }; ≡ int b[4] = { 7, 11, 100, 4 };
- int a[3][4] = { { 11, 32, 21, 250 },{ 211, 7, 162, 2 },{ 15, 180, 48, 190 }};

String

- Array of characters
  - char str [20];
  - char os[] = “UNIX”;
  - char os[5] = { ‘U’, ‘N’, ‘I’, ‘X’, ‘\0’};
- Built in string functions
  - #include <string.h>
  - strcpy(str, “csc215”);
  - i = strlen(str); /* i is 6 */

Enumeration

```
#include<stdio.h>
enum days {SUN, MON, TUE, WED, FRI,SAT};
main(){
   enum days day;
   day = MON;
   if (day == FRI || day == SAT)
        printf ("It’s the weekend!");
   else
        printf ("Lets try to work");
}
```

Enumeration

```
#include<stdio.h>
enum days {SUN, MON, TUE, WED, FRI,SAT};
 /* SUN = 0, M0N = 1 and so on*/
main(){
  enum days day;
   day = 1;  /* Same as day = MON */
   if (day == FRI || day == SAT)
        printf ("It's the weekend!");
   else
        printf ("Lets try to work");
}
```

Enumeration

```
#include<stdio.h>
enum days {SUN = 1, MON = 3, TUE, … SAT};
main(){
  enum days day;
  day = 4;
   if (day == FRI || day == SAT)
        printf ("It’s the weekend!");
   else
        printf ("Lets try to work!");
}
```