Programming Languages and Tools: Programming with C++ CS:3210:0003

Lecture/Lab #5

Operator Precedence

Operators	Precedence
!, +, – (unary operators)	first
*, /, %	second
+, –	third
<, <=, >=, >	fourth
==, !=	fifth
&&	sixth
||	seventh
= (assignment operator)	last

Operator Precedence

- int myNumber = 10 * 30 + 20 - 5 * 5;
- 10 * 30 + 20 - 5 * 5
- 300 + 20 - 5 * 5
- 300 + 20 - 25
- 320 - 25
- 295
- Use parentheses, this isn't worth the effort

Operators	Precedence
!, +, – (unary operators)	first
*, /, %	second
+, –	third
<, <=, >=, >	fourth
==, !=	fifth
&&	sixth
||	seventh
= (assignment operator)	last

Integer Number Systems

- Binary
- Octal
- Decimal
- Hexadecimal

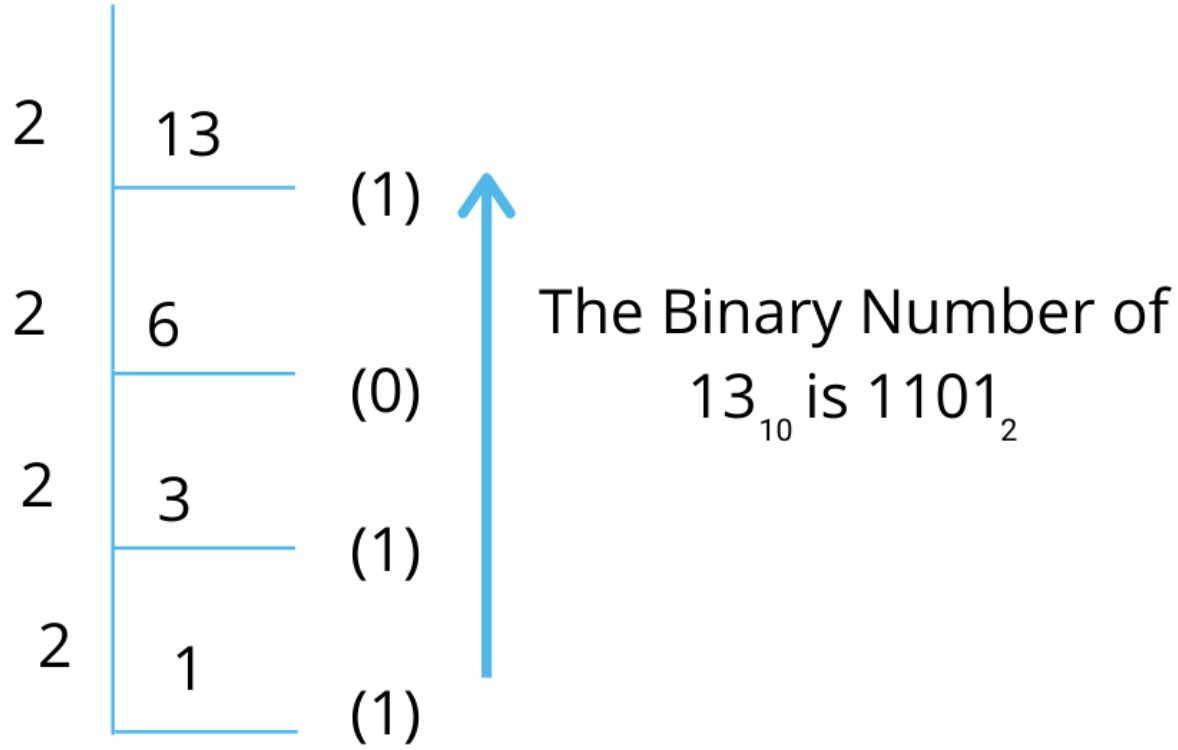
Decimal ⟷ Binary

- n binary bits can represent 2^n integer values
- Unsigned: 0 to 2^n-1
- Signed: $-2^{n/2}$-1 to +2n/2
- For example,
  - 4 bits can represent 2^4, that is, 16 integer values
  - For unsigned integers, 0 to 15
  - For signed integers -8 to 7

Decimal ⟷ Binary

Decimal Number	Binary equivalent	Decimal Number	Binary Equivalent
0	0000	8	1000
1	0001	9	1001
2	0010	10	1010
3	0011	11	1011
4	0100	12	1100
5	0101	13	1101
6	0110	14	1110
7	0111	15	1111

Decimal to Binary



Activity

Write a program called int2bin.cpp that inputs an integer between 0 and 15 and prints its binary representation in 4 bits

- Use only C++ types/constructs that we have covered so far
- So no loops

Compound Assignment

Operator	Usage	Equivalent
Addition Assignment	num1 += num2;	num1 = num1 + num2;
Subtraction Assignment	num1 -= num2;	num1 = num1 - num2;
Multiplication Assignment	num1 *= num2;	num1 = num1 * num2;
Division Assignment	num1 /= num2;	num1 = num1 / num2;
Modulo Assignment	num1 %= num2;	num1 = num1 % num2;

Functions

- Pass values to the function using arguments
- Function can return a value
- Syntax:

```
returnType functionName (arg1Type arg1Name, …, argnType argnName) {
        …
        return returnValue; // Must be of returnType
}
```

Static Arrays

- Collection of elements, all of the same type
- Syntax:

```
elementType arrayName [size];                  //declaration
elementType arrayName [size] = {initList};     //definition
```

- size must be a constant
- Examples:

```
int myNumbers [5] = {0};                       //all 5 elements are 0
char myChars [3] = {‘a’, ‘b’, ‘c’}
```

- Static – size is fixed at compile time

Static Arrays

- Compiler allocates contiguous memory for array elements
- Use index i to access the i+1th element of the array

```
int a [5] = {0};   //integer array of size 5
a[0] = 1;          //first element
a[4] = 5;          //last element
```

- Accessing index out of bounds of the array is a security bug
- Compiler cannot detect out-of-bounds access of an array
- Write programs that prevent this behavior

C++ Types

Type	Size (bits)	Size (bytes)	Range
char	8	1	-128 to 127
unsigned char	8	1	0 to 255
int	16	2	-2^{15} to $2^{15}-1$
unsigned int	16	2	0 to $2^{16}-1$
short int	8	1	-128 to 127
unsigned short int	8	1	0 to 255
long int	32	4	-2^{31} to $2^{31}-1$
unsigned long int	32	4	0 to $2^{32}-1$
float	32	4	3.4E-38 to 3.4E+38
double	64	8	1.7E-308 to 1.7E+308
long double	80	10	3.4E-4932 to 1.1E+4932