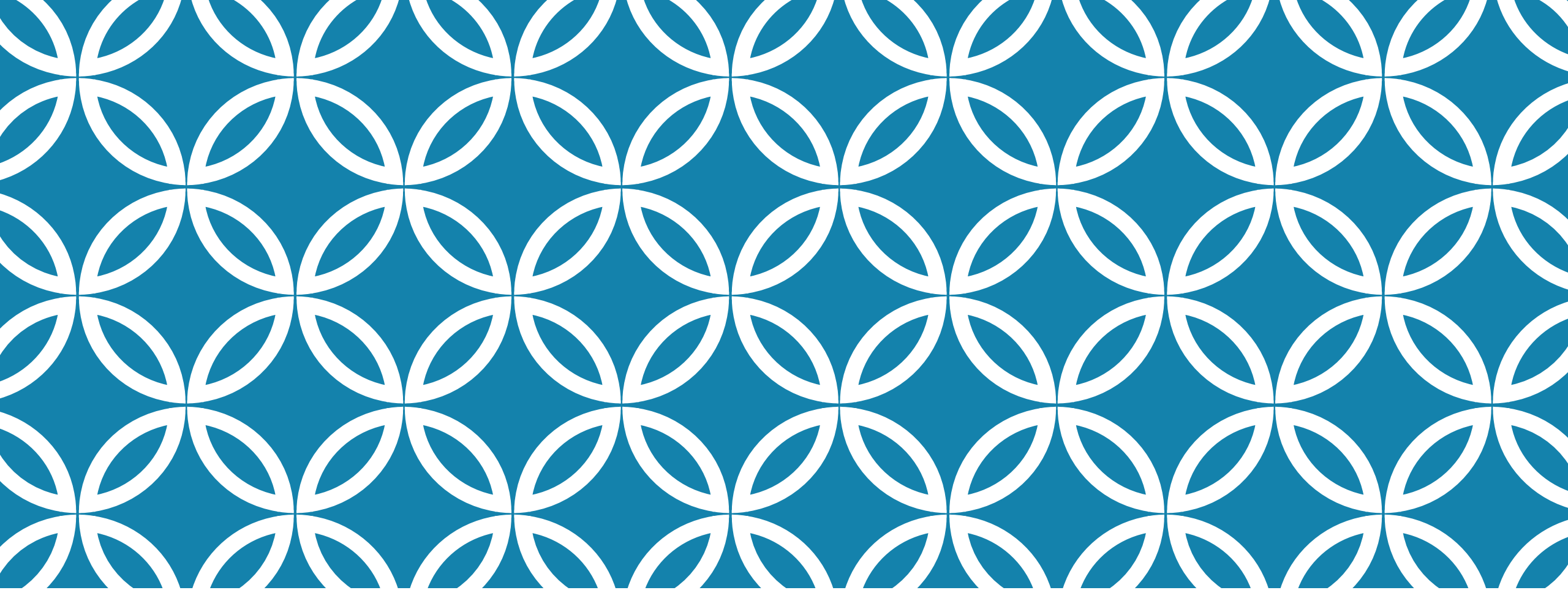

PAPER 102:
PROGRAMMING & PROBLEM SOLVING THROUGH C

Bidur Bhushan Handique
Dept. of Computer Science
St. Anthony's College

QUALIFIERS

- Qualifiers are keywords.
- They modify the properties of primary data types.

  Eg:

  int a; signed int a; unsigned int a;

- There are two types of qualifiers:
  - Size Qualifiers
  - Sign Qualifiers

SIZE QUALIFIERS

- Alters the size of basic data type (int, char, float, double)
- Two size qualifiers:
  - Short

    Min Size is 16 bits

    Eg: short int a;
  - Long

    Must be greater than or equal to an int, min size is 32 bits

    Eg: long int a;

    short int <= int <= long int
- Long can also be applied to double.

  Precision of long double>= precision of double >= precision of float

SIGN QUALIFIERS

- Sign qualifiers are of two types, and are applied to data types int and char:
  - Signed (takes both positive and negative integer)
  - Unsigned (takes only positive integer)
- Eg

```
unsigned int b;
signed int b;
```

SIZE OPERATOR

- The sizeof operator is treated as a unary operator.
- It gives the size of any data type in bytes.
- Example:

```
/* program to illustrate the sizeof operator*/
#include<stdio.h>
void main()
{       printf("size of short int is %d", sizeof(short int));
        printf("\n size of an int is %d", sizeof(int));
        printf("\n size of long int is %d", sizeof(long int));        }
```

DATA TYPES AND RANGE

Variable Type	Keyword	Bytes Required	Range
Character	char	1	-128 to 127
Unsigned character	unsigned char	1	0 to 255
Integer	int	2	-32768 to 32767
Short Integer	short int	2	-32768 to 32767
Long Integer	long int	4	-2,147,483,648 to 2,147,438,647
Unsigned Integer	unsigned int	2	0 to 65535
Unsigned Short Integer	unsigned short int	2	0 to 65535
Unsigned Long Integer	unsigned long int	4	0 to 4,294,967,295
Float	float	4	1.2E-38 to
Double	double	8	2.2E-308 to
Long Double	long double	10	3.4E-4932 to 1.1E+4932

CONST QUALIFIER

- Const qualifier, prevent the value from being changed, through out the execution of a program.
- And so it can be modified only in the declaration part.
- Eg

```
const float pi=3.14;
```

BITWISE OPERATOR

- A bitwise operator operates on each bit of data.
- These operators are used for testing, complementing or shifting bits to the right or left.
- They are meant for integers only.
- A list of bitwise operators:

Operators	Meaning
&	AND
|	OR
^	XOR
<<	Shift Left
>>	Shift Right
~	Bitwise Complement

BITWISE AND

- Eg: int a=13, b=7, c; c=a & b;
- Assuming int is 16 bit.
- Bitwise AND operator (&)

a 0000 0000 0000 1101 (13)

b 0000 0000 0000 0111 (7)

c 0000 0000 0000 0101 (5)

BITWISE OR

- Eg: int a=13, b=7, c; c=a | b;
- Assuming int is 16 bit.
- Bitwise OR operator (|)

a 0000 0000 0000 1101 (13)

b 0000 0000 0000 0111 (7)

c 0000 0000 0000 1111 (15)

BITWISE XOR

- Eg: int a=13, b=7, c; c=a ^ b;
- Assuming int is 16 bit.
- Bitwise AND operator (^)

a 0000 0000 0000 1101 (13)

b 0000 0000 0000 0111 (7)

c 0000 0000 0000 1010 (10)

SHIFT LEFT

- Eg: int a=13, c; c=a << 3;

a 0000 0000 0000 1101 (13)

(Drop ←) (Insert Zero ←)

c 0000 0000 0110 1000 (104)

SHIFT RIGHT

- Eg: int a=13, c; c=a >> 2;

a 0000 0000 0000 1101 (13)

(Insert Zero →) (Drop →)

c 0000 0000 0000 0011(3)

SHIFTING NEGATIVE NUMBERS

- Eg: int a=-5, c; c=a >> 2;

a 0000 0000 0000 0101 (5)

One Complement

1111 1111 1111 1010

Two Complement

\+ 1

1111 1111 1111 1011

(Binary representation of -5)

Since, c=a>>2;//right shift

(Insert One→) (Drop →)

c 1111 1111 1111 1110

One Complement

0000 0000 0000 0001

Two Complement

\+ 1

0000 0000 0000 0010 (-2)