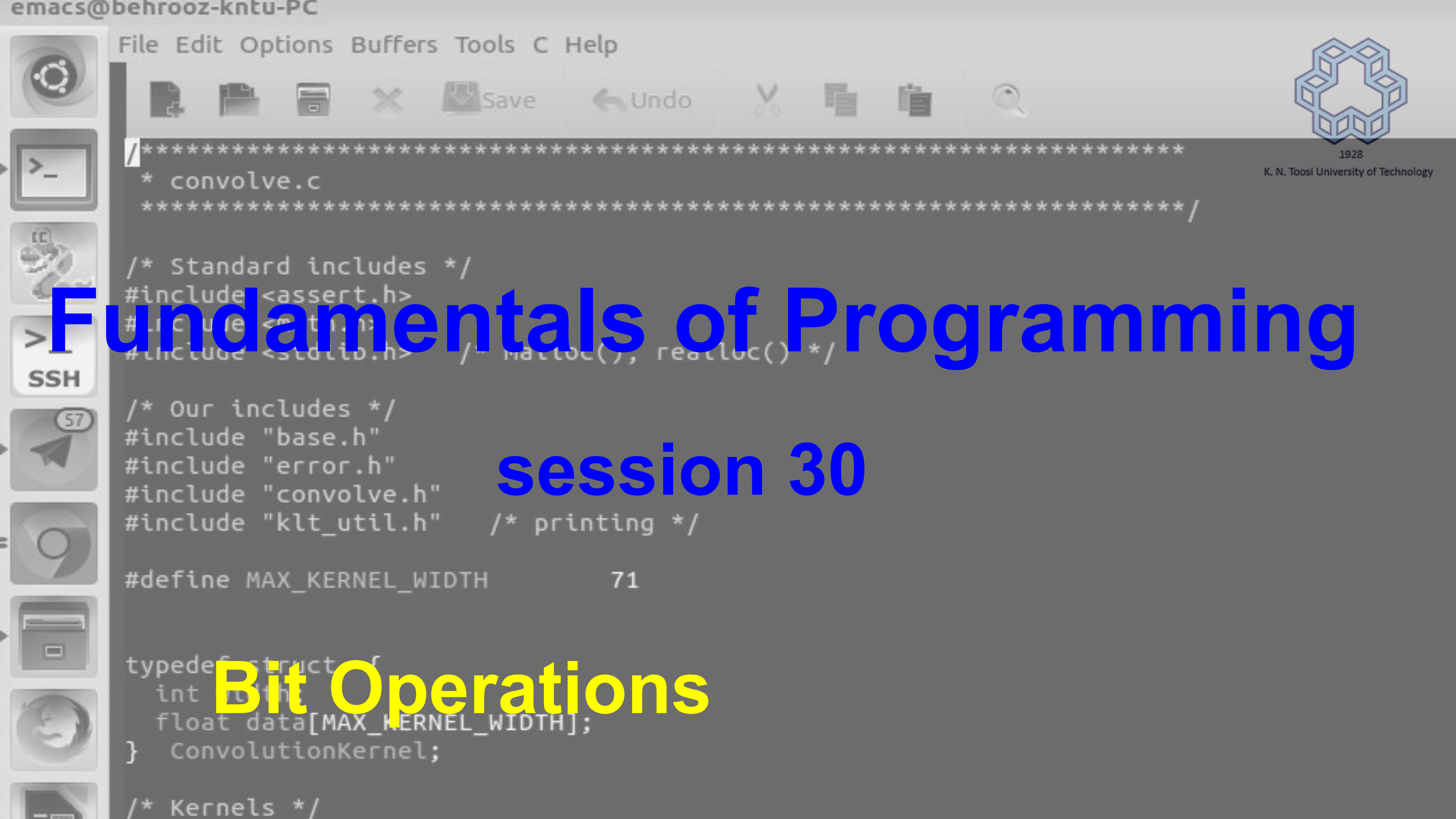
Fundamentals of Programming

session 30

Bit Operations

1928
K. N. Toosi University of Technology



```
/*********************************************************************
 * convolve.c
 *********************************************************************/

/* Standard includes */
#include <assert.h>
#include <math.h>
#include <stdlib.h>		/* malloc(), realloc() */

/* Our includes */
#include "base.h"
#include "error.h"
#include "convolve.h"
#include "klt_util.h"   /* printing */

#define MAX_KERNEL_WIDTH 	71


typedef struct {
  int width;
  float data[MAX_KERNEL_WIDTH];
}  ConvolutionKernel;

/* Kernels */
```

Bitwise operations

- AND (&)
- OR (|)
- XOR (^)
- NOT (~)

Bitwise AND

x	y	x & y
0	0	0
0	1	0
1	0	0
1	1	1

Bitwise OR

1928
K. N. Toosi University of Technology

`x`	`y`	`x	y`
0	0	0
0	1	1
1	0	1
1	1	1

Bitwise XOR

`x`	`y`	`x ^ y`
0	0	0
0	1	1
1	0	1
1	1	0

Bitwise NOT

`x`	`~x`
0	1
1	0

Bitwise Operations

```
char x = 0x6D,y = 0x8E;
```

`x`	01101101
`y`	10001110
`x & y`	00001100
`x | y`	11101111
`x ^ y`	11100011
`~x`	10010010

Shift operations

1928
K. N. Toosi University of Technology

```
unsigned char x = 0x6D;
```

`x`	01101101
`x << 1`	11011010
`x << 3`	01101000

Shift operations

1928
K. N. Toosi University of Technology

```
unsigned char x = 0x6D;
```

`x`	01101101
`x << 1`	11011010
`x << 3`	01101000

Shift operations

```
unsigned short int x = 0x6D;
```

x	00000000 01101101	109
x << 1	00000000 11011010	218
x << 3	00000011 01101000	872

Shift operations

1928
K. N. Toosi University of Technology

```
unsigned char x = 0xED;
```

x	11101101	237	
x >> 1	01110110	118	= 237/2
x >> 3	00011101	29	= 237/8

Shift operations

1928
K. N. Toosi University of Technology

```
signed char x = 0xED, y = 0x6D;
```

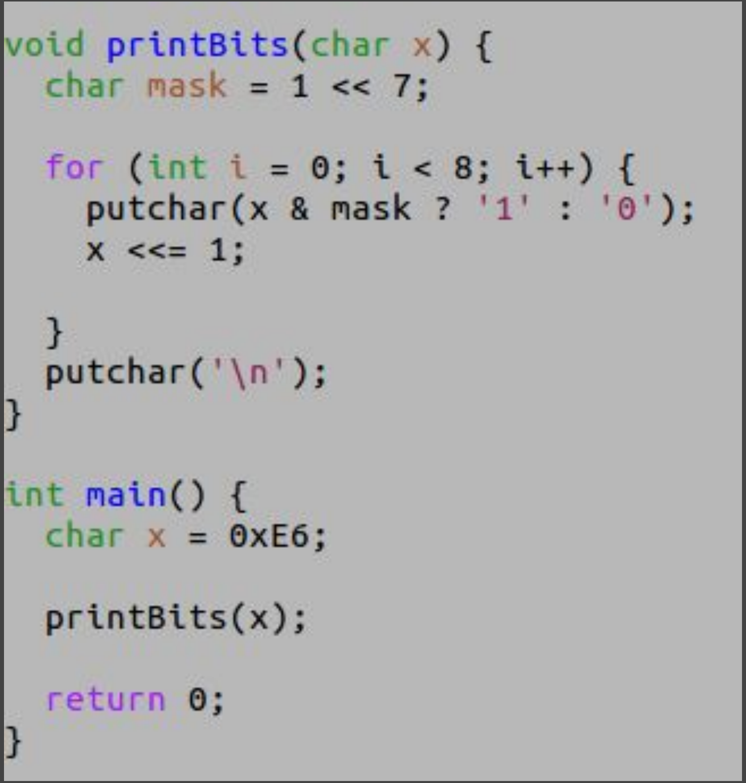
x	11101101	-19
x >> 1	11110110	-10
x >> 3	11111101	-3
y	01101101	109
y >> 1	00110110	54
y >> 3	00001101	13

Shift operations

1928
K. N. Toosi University of Technology

`x &= y`	`x = x & y`
`x |= y`	`x = x | y`
`x ^= y`	`x = x ^ y`
`x <<= y`	`x = x << y`
`x >>= y`	`x = x >> y`

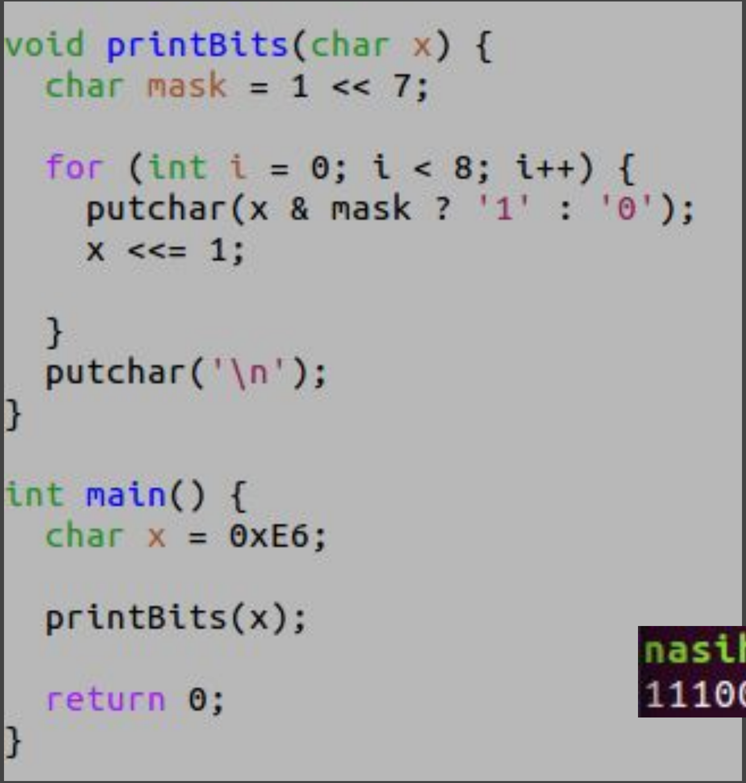
Printing numbers in bits



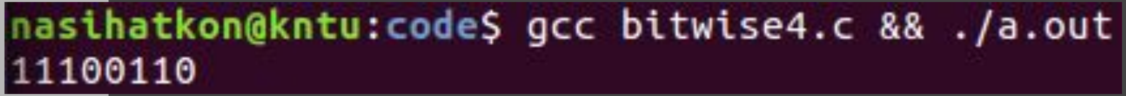
```c
void printBits(char x) {
  char mask = 1 << 7;

  for (int i = 0; i < 8; i++) {
    putchar(x & mask ? '1' : '0');
    x <<= 1;

  }
  putchar('\n');
}

int main() {
  char x = 0xE6;

  printBits(x);

  return 0;
}
```

Printing numbers in bits

1928
K. N. Toosi University of Technology



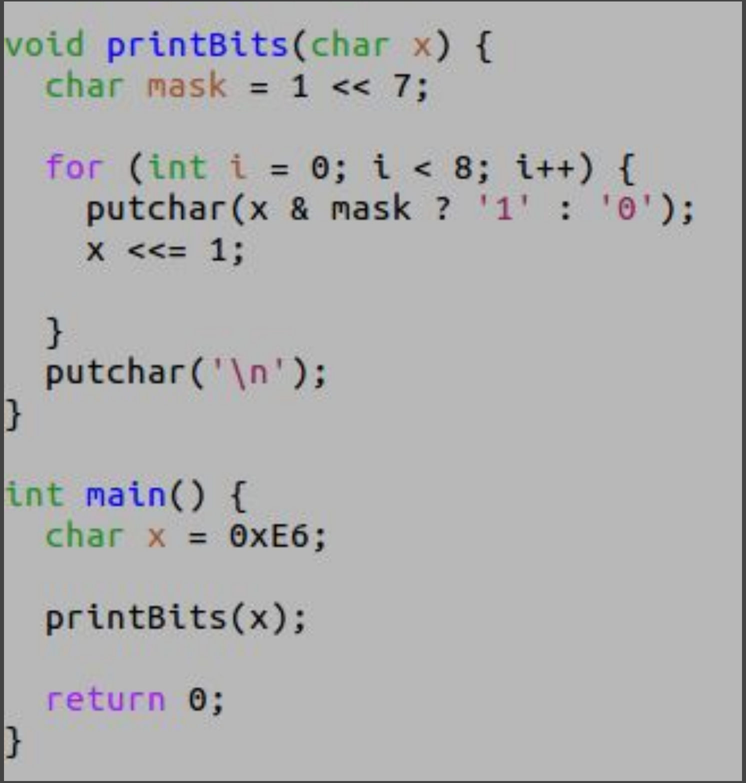
```c
void printBits(char x) {
  char mask = 1 << 7;

  for (int i = 0; i < 8; i++) {
    putchar(x & mask ? '1' : '0');
    x <<= 1;

  }
  putchar('\n');
}

int main() {
  char x = 0xE6;

  printBits(x);

  return 0;
}
```



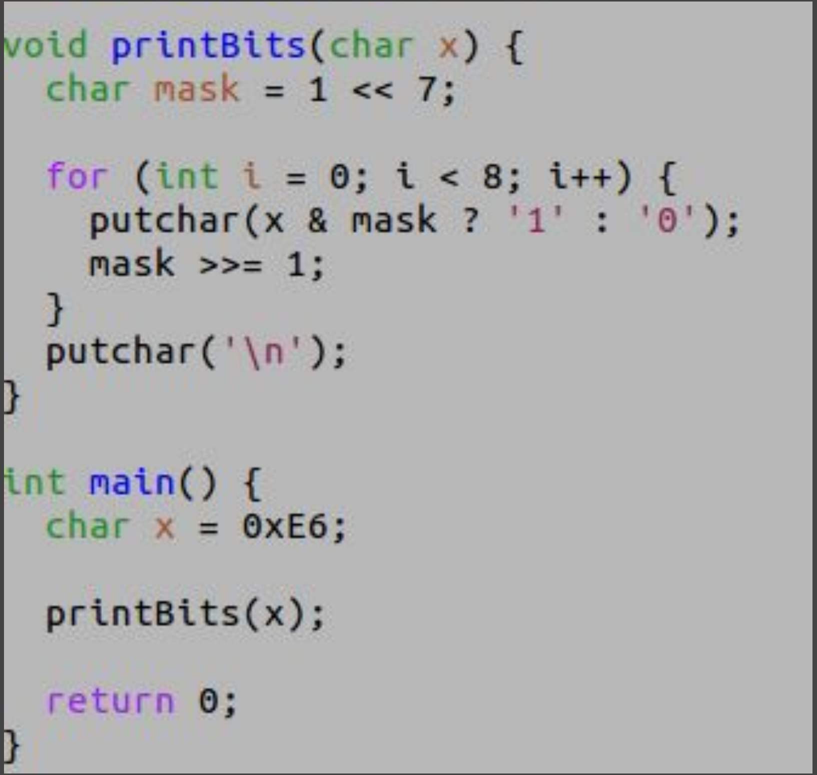
```
nasihatkon@kntu:code$ gcc bitwise4.c && ./a.out
11100110
```

Printing numbers in bits

1928
K. N. Toosi University of Technology



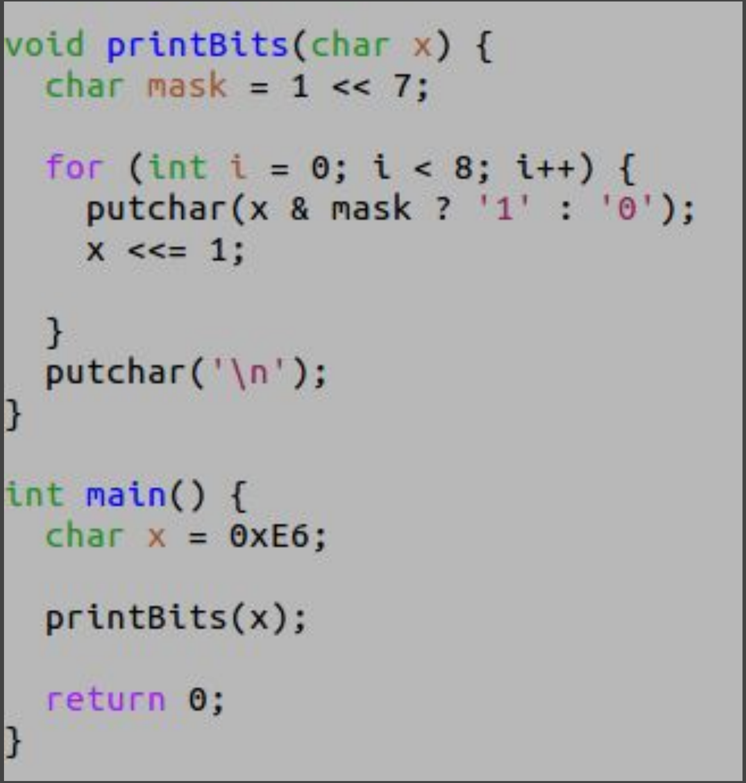
```c
void printBits(char x) {
  char mask = 1 << 7;

  for (int i = 0; i < 8; i++) {
    putchar(x & mask ? '1' : '0');
    x <<= 1;

  }
  putchar('\n');
}

int main() {
  char x = 0xE6;

  printBits(x);

  return 0;
}
```



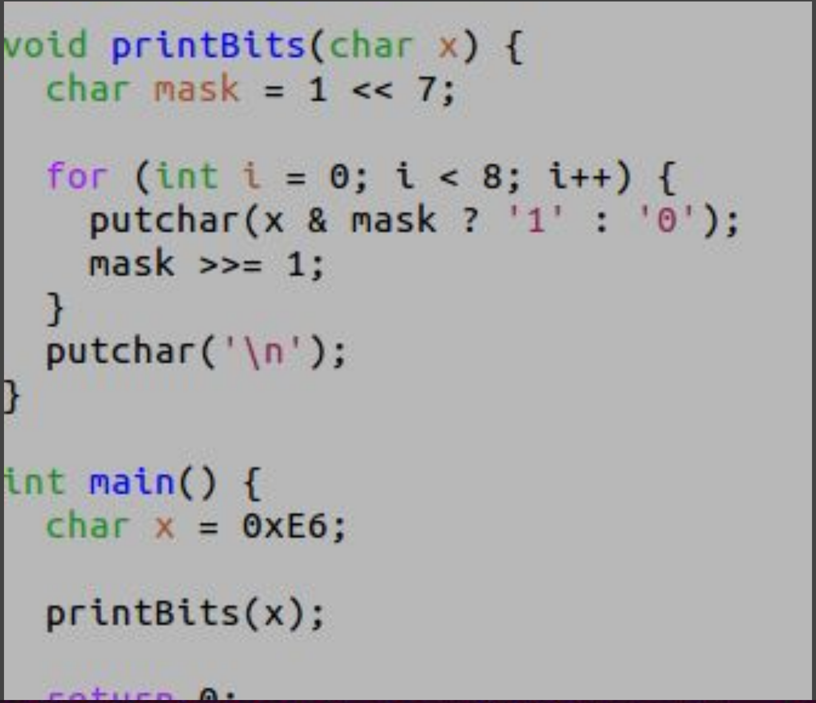
```c
void printBits(char x) {
  char mask = 1 << 7;

  for (int i = 0; i < 8; i++) {
    putchar(x & mask ? '1' : '0');
    mask >>= 1;
  }
  putchar('\n');
}

int main() {
  char x = 0xE6;

  printBits(x);

  return 0;
}
```

Printing numbers in bits

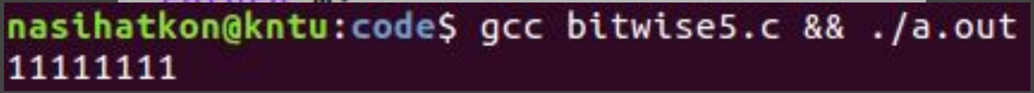
1928
K. N. Toosi University of Technology



```c
void printBits(char x) {
  char mask = 1 << 7;

  for (int i = 0; i < 8; i++) {
    putchar(x & mask ? '1' : '0');
    x <<= 1;

  }
  putchar('\n');
}

int main() {
  char x = 0xE6;

  printBits(x);

  return 0;
}
```



```c
void printBits(char x) {
  char mask = 1 << 7;

  for (int i = 0; i < 8; i++) {
    putchar(x & mask ? '1' : '0');
    mask >>= 1;
  }
  putchar('\n');
}

int main() {
  char x = 0xE6;

  printBits(x);

  return 0;
```



```
nasihatkon@kntu:code$ gcc bitwise5.c && ./a.out
11111111
```

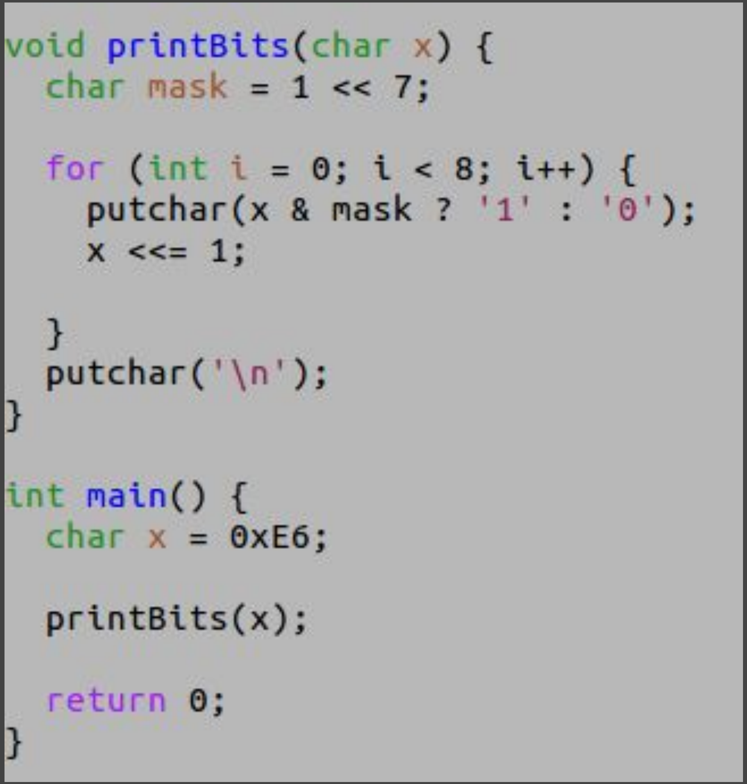
Printing numbers in bits

```c
void printBits(char x) {
  char mask = 1 << 7;

  for (int i = 0; i < 8; i++) {
    putchar(x & mask ? '1' : '0');
    x <<= 1;

  }
  putchar('\n');
}

int main() {
  char x = 0xE6;

  printBits(x);

  return 0;
}
```

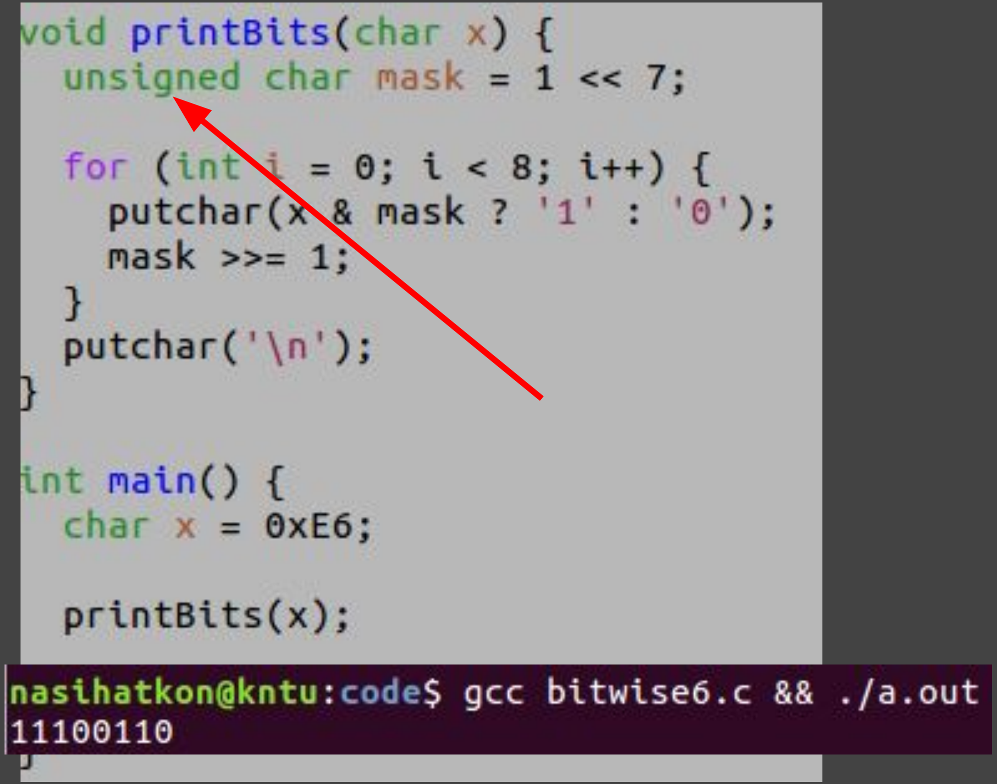
```c
void printBits(char x) {
  unsigned char mask = 1 << 7;

  for (int i = 0; i < 8; i++) {
    putchar(x & mask ? '1' : '0');
    mask >>= 1;
  }
  putchar('\n');
}

int main() {
  char x = 0xE6;

  printBits(x);

  return 0;
}
```

1928
K. N. Toosi University of Technology

Printing numbers in bits



```c
void printBits(char x) {
  char mask = 1 << 7;

  for (int i = 0; i < 8; i++) {
    putchar(x & mask ? '1' : '0');
    x <<= 1;

  }
  putchar('\n');
}

int main() {
  char x = 0xE6;

  printBits(x);

  return 0;
}
```



```c
void printBits(char x) {
  unsigned char mask = 1 << 7;

  for (int i = 0; i < 8; i++) {
    putchar(x & mask ? '1' : '0');
    mask >>= 1;
  }
  putchar('\n');
}

int main() {
  char x = 0xE6;

  printBits(x);
```

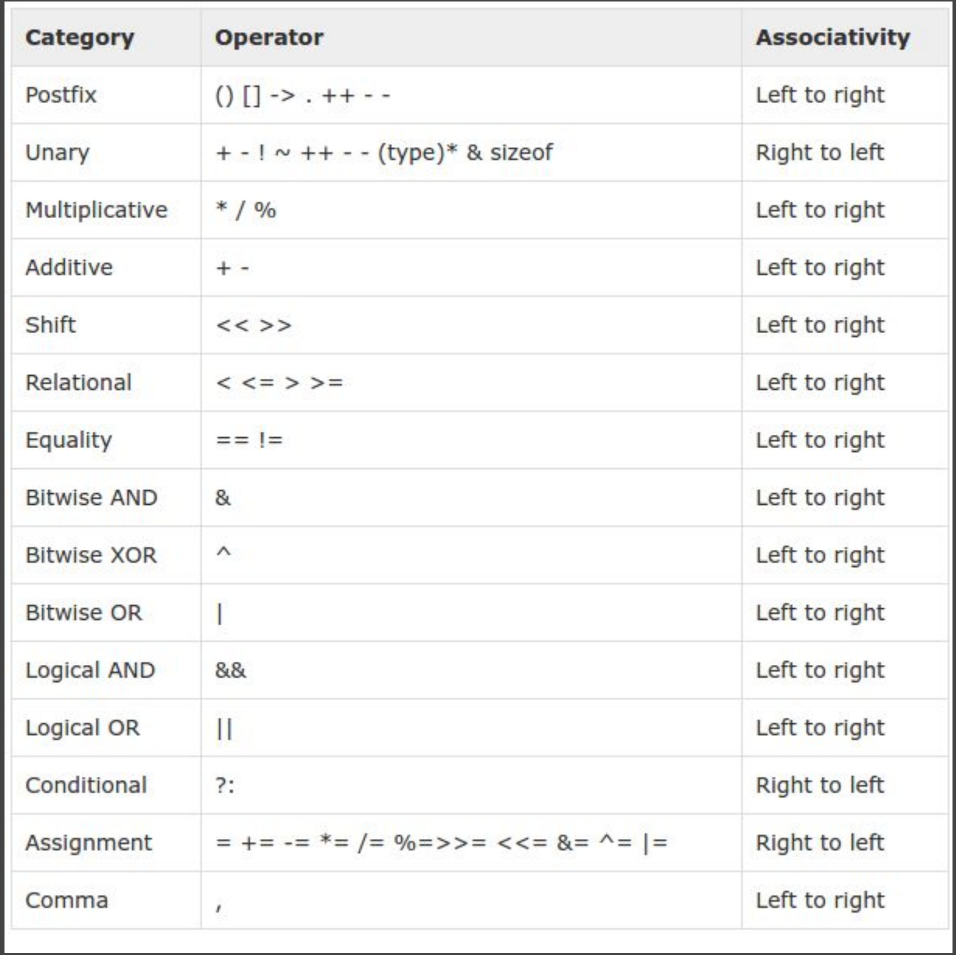
```
nasihatkon@kntu:code$ gcc bitwise6.c && ./a.out
11100110
```

mind precedence

1928
K. N. Toosi University of Technology

```
if ( x & mask == 0 )
```

```
if ( (x & mask) == 0 )
```

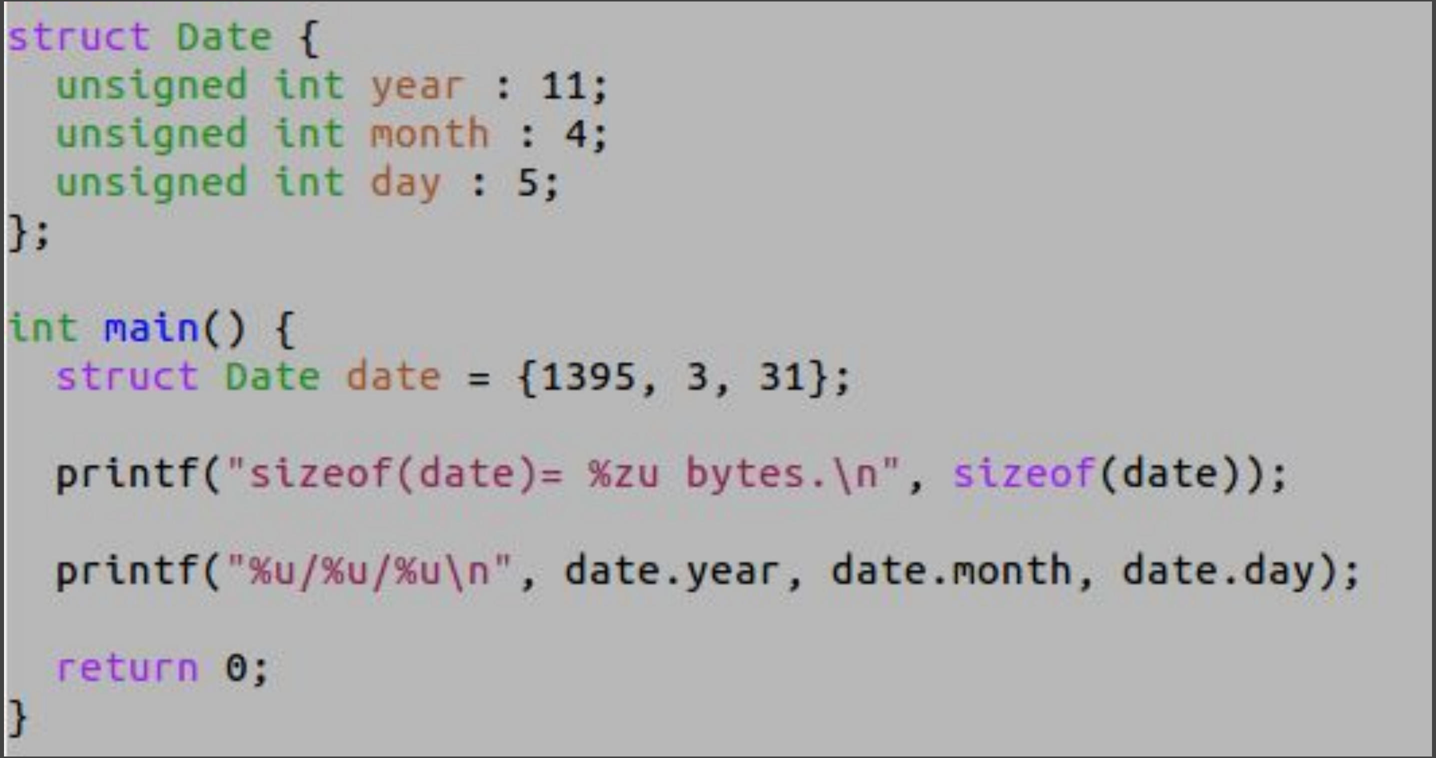


Category	Operator	Associativity
Postfix	() [] -> . ++ - -	Left to right
Unary	+ - ! ~ ++ - - (type)* & sizeof	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Shift	<< >>	Left to right
Relational	< <= > >=	Left to right
Equality	== !=	Left to right
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR	|	Left to right
Logical AND	&&	Left to right
Logical OR	||	Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %=>>= <<= &= ^= |=	Right to left
Comma	,	Left to right

https://www.tutorialspoint.com/cprogramming/c_operators_precedence.htm

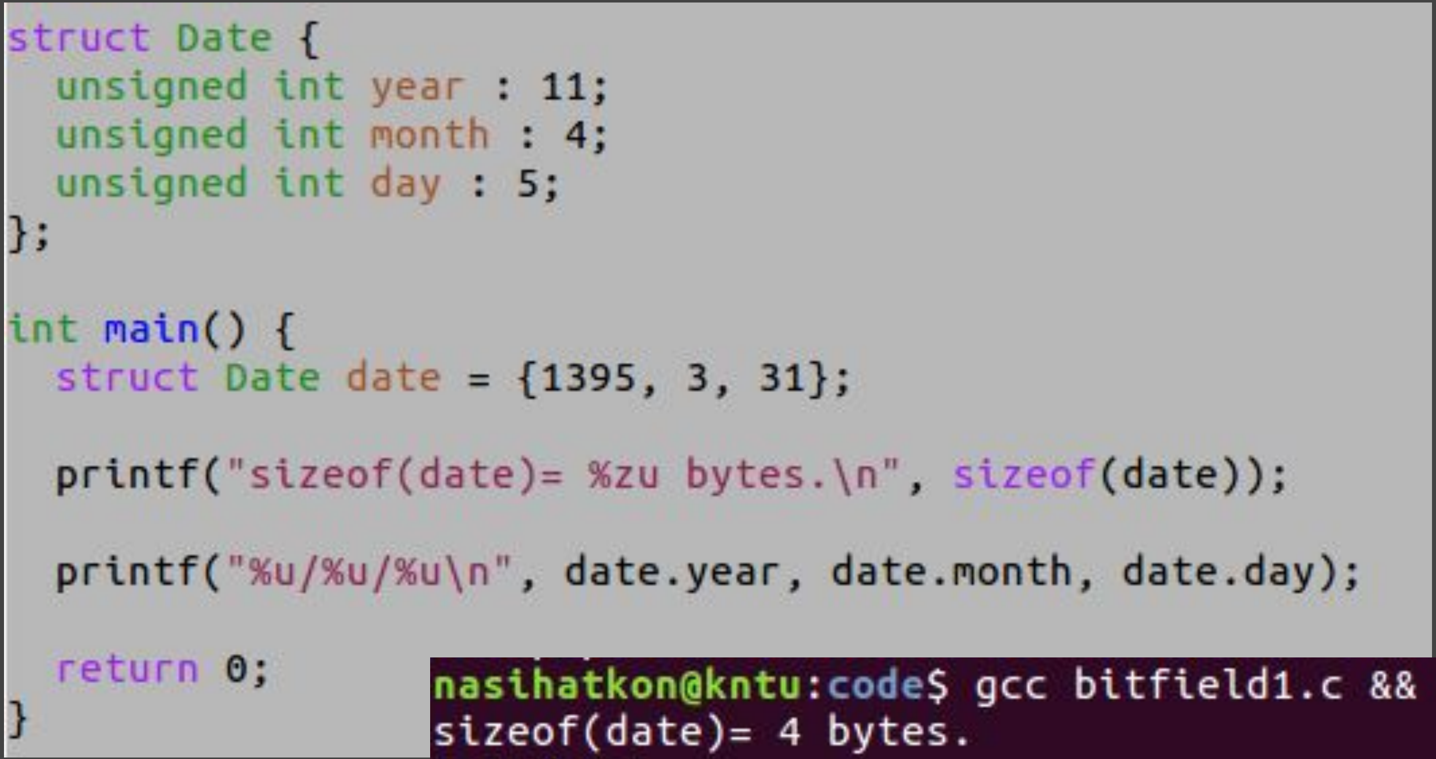
Bit Fields

1928
K. N. Toosi University of Technology

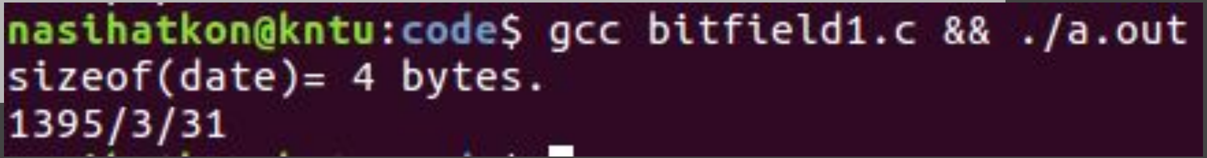


```
struct Date {
  unsigned int year : 11;
  unsigned int month : 4;
  unsigned int day : 5;
};

int main() {
  struct Date date = {1395, 3, 31};

  printf("sizeof(date)= %zu bytes.\n", sizeof(date));

  printf("%u/%u/%u\n", date.year, date.month, date.day);

  return 0;
}
```

Bit Fields



```c
struct Date {
  unsigned int year : 11;
  unsigned int month : 4;
  unsigned int day : 5;
};

int main() {
  struct Date date = {1395, 3, 31};

  printf("sizeof(date)= %zu bytes.\n", sizeof(date));

  printf("%u/%u/%u\n", date.year, date.month, date.day);

  return 0;
}
```



```
nasihatkon@kntu:code$ gcc bitfield1.c && ./a.out
sizeof(date)= 4 bytes.
1395/3/31
```