CS 24000 - Programming In C

Week 12: Review of
type conversion

Zhiyuan Li
Department of Computer Science
Purdue University, USA

Type conversion

- From unsigned char to int
- From signed char to int
- The result of byte addition and subtraction
  - Unsigned versus signed
  - Assigning to a char versus writing to an int

Unsigned char to int

- Unsigned char is viewed as a nonnegative int
- Hence, when promoted to int, 0's will be padded to the more significant bytes

```
#include <stdio.h>

int main() {
 int a;
 unsigned char b= 0x80;
 a = b;
printf("%x\n", a);
printf("%d\n", a);

}
```

```
80
128
```

Signed char to int

- The sign bit will be extended to the higher bits
- A nonnegative signed char has the sign bit 0
- when promoted int will have 0's padded to the more significant bytes

```
#include <stdio.h>

int main() {
 int a;
 char b= 0x48;
 a = b;
printf("%x\n", a);
printf("%d\n", a);
}
```

48

72

Signed char to int

- A negative char has sign bit 1
- when promoted int will have 1's padded to the more significant bytes

```
#include <stdio.h>

int main() {
 int a;
 char b= 0x80;
 a = b;
printf("%x\n", a);
printf("%d\n", a);
}
```

ffffff80

-128

Int to unsigned char

- truncate

```
#include <stdio.h>

int main() {
 int a = 0x1b0;
unsigned char b;

 b = a;
 printf("%x\n", b);
 printf("%c\n", b);
 printf("%d\n", b);
 return 0;
}
```

```
b0
°
176
```

Int to unsigned char (cont'd)

```
#include <stdio.h>

int main() {
 int a = 0xf010;
unsigned char b;

 b = a;
 printf("%x\n", b);
 printf("%c\n", b);
 printf("%d\n", b);
 return 0;
}
```

10

16

Int to signed char

- Truncate

```
#include <stdio.h>
int main() {
 int a = 0x1b0;
 char b;
 unsigned char c;
 b = a;
 printf("%x\n", b);
 printf("%c\n", b);
 printf("%d\n", b);
 c = (unsigned char) b;  // same w/o (unsigned char)
 printf("%x\n", c);
 printf("%c\n", c);
 printf("%d\n", c);
 return 0;
}
```

```
fffffb0
°
-80
b0
°
176
```

Review and explain byte add/subtract

- Promote to int before add/subtract
- Unsigned byte add/subtract
- Signed byte add/subtract
- Write to char (truncate)
- Write to unsigned char (truncate)
- Write to int (no truncate!)

```
#include <stdio.h>

main() {

        unsigned char ua = 0, ub = -1, ux;
        ux = ua + ub;
        printf("%X\n", ux);
        printf( %d\n", ux);
        return 0;
}
```

ux is 0XFF

```
#include <stdio.h>

main() {

unsigned char uc = 0, ud =
1, uy;
 uy = uc - ud;
 printf(" %X\n", uy);
 return 0;
}
```

uy is 0XFF

```
#include <stdio.h>

main() {

        unsigned char ua = 0, ub = -1, ux;
        unsigned char uc = 0, ud =  1, uy;
        int w, z;
        w  = ua + ub;
        z  = uc - ud;
        printf("%X\n", w);
        printf("%X\n", z);
        return 0;
}
```

w is 0XFF

z is 0XFFFFFFFF

```
#include <stdio.h>

main() {

        unsigned char ua = 0, ub = -1, ux, uf;
        unsigned char uc = 0, ud =  1, uy;
        int  z;
        uf = uc – ud;
        z  = uf;
        printf("%X\n", z);
        return 0;
}
```

z is OXFF

Question from Quiz 6

```
#include <stdio.h>

int main() {

char a = '\xff';
int    x;
x  =  a << 1;
printf("%#X\n", x);

return 0;
}
```

0XFFFFFFFE

Another left-shift example for signed char

```
#include <stdio.h>

main() {
        char a = '\x80';
        unsigned int x;
        x = a << 1;
        printf("x is \t %#X\n",x);
        return 0;
}
```

x is 0XFFFFFF00

- Lesson: cast to unsigned char before left shifting unless wanting to use left shift to multiply a number by 2

Right-shift example for signed char

```
#include <stdio.h>

main() {
        char a = '\x80';
        unsigned int x;
        x = a >> 1;
        printf("x is \t %#X\n",x);
        return 0;
}
```

x is 0XFFFFFFC0

- Lesson: cast to unsigned char before shifting unless using right shift to do divide by 2

A bad case of arithmetic right shift

```
#include <stdio.h>

main() {

        char a = '\xff';
        unsigned int x;

        x = a >> 1;

        printf("x is \t %#X\n",x);

        return 0;
}
```

x is 0XFFFFFFFF

Quiz 8 #1

```
#include <stdio.h>

main() {
 unsigned char ubig = '\xff‘;;
 int ioverflow;
 ioverflow = ubig + ubig;
 printf("%X\n", ioverflow);
 return 0;
|
```

- What does the program above print (assume 32-bit integer)?
- (a) FF
- (b) FE
- (c) 1FE
- (d) none of the above

- Answer (c) 1FE

Quiz 8 #2

```
#include <stdio.h>

main() {
 char ubig = '\xff';;
 int ioverflow;
 ioverflow = ubig + ubig;
 printf("%X\n", ioverflow);
 return 0;
}
```

- What does the program above print (assume 32 bit integer)?
- (a) FE
- (b) FFFFFFFE
- (c) 1FE
- (d) 000001FE
- (e) none of the above

Answer

- (b) FFFFFFFE

```
include <stdio.h>
main() {
        char a = '\x81', b;
        int x;
        x = a + a;
        printf("%X\n", x);
        return 0;
}
```

- What does the program above print (assuming 32-bit integer)?
- (a) 00000162
- (b) FFFFFF02
- (c) 00000002
- (d) 2
- (e) none of the above

Answer

- (b) FFFFFF02

Continue with processes