

## binary.c

1/1

lectures/7/src/

```

1: /*****
2:  * binary.c
3:  *
4:  * Computer Science 50
5:  * David J. Malan
6:  *
7:  * Displays a number in binary.
8:  *
9:  * Demonstrates bitwise operators.
10: *****/
11:
12: #include <cs50.h>
13: #include <stdio.h>
14:
15:
16: int
17: main(void)
18: {
19:     // prompt user for number
20:     int n;
21:     do
22:     {
23:         printf("Non-negative integer please: ");
24:         n = GetInt();
25:     }
26:     while (n < 0);
27:
28:     // print number in binary
29:     for (int i = sizeof(int) * 8 - 1; i >= 0; i--)
30:     {
31:         int mask = 1 << i;
32:         if (n & mask)
33:             printf("1");
34:         else
35:             printf("0");
36:     }
37:     printf("\n");
38: }

```

## bmp.h

1/2

lectures/7/src/

```

1: /*****
2:  * bmp.h
3:  *
4:  * Computer Science 50
5:  * Problem Set 5
6:  *
7:  * BMP-related data types based on Microsoft's own.
8:  *****/
9:
10: #include <stdint.h>
11:
12:
13: /*
14:  * Common Data Types
15:  *
16:  * The data types in this section are essentially aliases for C/C++
17:  * primitive data types.
18:  *
19:  * Adapted from http://msdn.microsoft.com/en-us/library/cc230309\(PROT.10\).aspx.
20:  * See http://en.wikipedia.org/wiki/Stdint.h for more on stdint.h.
21:  */
22:
23: typedef uint8_t  BYTE;
24: typedef uint32_t DWORD;
25: typedef int32_t  LONG;
26: typedef uint16_t WORD;
27:
28:
29: /*
30:  * BITMAPFILEHEADER
31:  *
32:  * The BITMAPFILEHEADER structure contains information about the type, size,
33:  * and layout of a file that contains a DIB [device-independent bitmap].
34:  *
35:  * Adapted from http://msdn.microsoft.com/en-us/library/dd183374\(VS.85\).aspx.
36:  */
37:
38: typedef struct
39: {
40:     WORD    bfType;
41:     DWORD   bfSize;
42:     WORD    bfReserved1;
43:     WORD    bfReserved2;
44:     DWORD   bfOffBits;
45: } __attribute__((__packed__))
46: BITMAPFILEHEADER;
47:
48:
49: /*
50:  * BITMAPINFOHEADER
51:  *
52:  * The BITMAPINFOHEADER structure contains information about the
53:  * dimensions and color format of a DIB [device-independent bitmap].
54:  *
55:  * Adapted from http://msdn.microsoft.com/en-us/library/dd183376\(VS.85\).aspx.
56:  */
57:
58: typedef struct
59: {
60:     DWORD   biSize;
61:     LONG    biWidth;
62:     LONG    biHeight;
63:     WORD    biPlanes;
64:     WORD    biBitCount;

```

## bmp.h

lectures/7/src/

2/2

```

65:     DWORD biCompression;
66:     DWORD biSizeImage;
67:     LONG biXPelsPerMeter;
68:     LONG biYPelsPerMeter;
69:     DWORD biClrUsed;
70:     DWORD biClrImportant;
71: } __attribute__((__packed__))
72: BITMAPINFOHEADER;
73:
74:
75: /*
76: * RGBTRIPLE
77: *
78: * This structure describes a color consisting of relative intensities of
79: * red, green, and blue.
80: *
81: * Adapted from http://msdn.microsoft.com/en-us/library/aa922590.aspx.
82: */
83:
84: typedef struct
85: {
86:     BYTE rgbtBlue;
87:     BYTE rgbtGreen;
88:     BYTE rgbtRed;
89: } __attribute__((__packed__))
90: RGBTRIPLE;

```

## endian.c

lectures/7/src/

1/2

```

1: /*****
2:  * endian.c
3:  *
4:  * Computer Science 50
5:  * David J. Malan
6:  *
7:  * Reads bf.bfSize from a BMP.
8:  *
9:  * Demonstrates endianness.
10: *****/
11:
12: #include <stdint.h>
13: #include <stdio.h>
14: #include <stdlib.h>
15:
16:
17: int
18: main(int argc, char *argv[])
19: {
20:     // ensure proper usage
21:     if (argc != 2)
22:         return 1;
23:
24:     // open file
25:     FILE *fp = fopen(argv[1], "r");
26:     if (fp == NULL)
27:         return 1;
28:
29:     // seek to BITMAPFILEHEADER's bfSize
30:     fseek(fp, 2, SEEK_SET);
31:
32:     // read in BITMAPFILEHEADER's bfSize
33:     uint32_t bfSize;
34:     fread(&bfSize, sizeof(bfSize), 1, fp);
35:
36:     // print bfSize
37:     printf("\nbfSize: %d\n\n", bfSize);
38:
39:     // return to start of file
40:     rewind(fp);
41:
42:     // read in BITMAPFILEHEADER's raw bytes
43:     uint8_t *buffer = malloc(14);
44:     fread(buffer, 1, 14, fp);
45:
46:     // print field via cast
47:     printf("bfSize: %d\n\n", *((uint32_t *) (buffer + 2)));
48:
49:     // print individual bytes in decimal
50:     printf("bfSize: %d %d %d %d\n",
51:         buffer[2], buffer[3], buffer[4], buffer[5]);
52:
53:     // print individual bytes in hexadecimal
54:     printf("bfSize: 0x%x 0x%x 0x%x 0x%x\n",
55:         buffer[2], buffer[3], buffer[4], buffer[5]);
56:
57:     // print individual bytes in binary
58:     printf("bfSize: ");
59:     for (int i = 2; i < 6; i++)
60:     {
61:         for (int j = 7; j >= 0; j--)
62:         {
63:             int mask = 1 << j;
64:             if (buffer[i] & mask)

```

**endian.c**

2/2

lectures/7/src/

```
65:         printf("1");
66:     else
67:         printf("0");
68: }
69: }
70: printf("\n\n");
71:
72: // that's all folks
73: return 0;
74: }
```

**memory.c**

1/1

lectures/7/src/

```
1: /*****
2:  * memory.c
3:  *
4:  * Computer Science 50
5:  * David J. Malan
6:  *
7:  * Demonstrates memory-related errors.
8:  *
9:  * problem 1: heap block overrun
10:  * problem 2: memory leak -- x not freed
11:  *
12:  * Adapted from
13:  * http://valgrind.org/docs/manual/quick-start.html#quick-start.prepare.
14:  *****/
15:
16: #include <stdlib.h>
17:
18:
19: void
20: f(void)
21: {
22:     int *x = malloc(10 * sizeof(int));
23:     x[10] = 0;
24: }
25:
26:
27: int
28: main(void)
29: {
30:     f();
31:     return 0;
32: }
```

pointers1.c  
lectures/7/src/

1/1

```
1: /*****
2:  * pointers1.c
3:  *
4:  * Computer Science 50
5:  * David J. Malan
6:  *
7:  * Prints a string, one character per line.
8:  *
9:  * Demonstrates strings as arrays.
10: *****/
11:
12: #include <cs50.h>
13: #include <stdio.h>
14: #include <string.h>
15:
16:
17: int
18: main(void)
19: {
20:     // prompt user for string
21:     printf("String please: ");
22:     char *s = GetString();
23:     if (s == NULL)
24:         return 1;
25:
26:     // print string, one character per line
27:     for (int i = 0, n = strlen(s); i < n; i++)
28:         printf("%c\n", s[i]);
29:
30:     // free string
31:     free(s);
32:
33:     return 0;
34: }
```

pointers2.c  
lectures/7/src/

1/1

```
1: /*****
2:  * pointers2.c
3:  *
4:  * Computer Science 50
5:  * David J. Malan
6:  *
7:  * Prints a string, one character per line.
8:  *
9:  * Demonstrates pointer arithmetic.
10: *****/
11:
12: #include <cs50.h>
13: #include <stdio.h>
14: #include <string.h>
15:
16:
17: int
18: main(void)
19: {
20:     // prompt user for string
21:     printf("String please: ");
22:     char *s = GetString();
23:     if (s == NULL)
24:         return 1;
25:
26:     // print string, one character per line
27:     for (int i = 0, n = strlen(s); i < n; i++)
28:         printf("%c\n", *(s+i));
29:
30:     // free string
31:     free(s);
32:
33:     return 0;
34: }
```

## swap2.c

1/1

lectures/7/src/

```
1: /*****
2:  * swap2.c
3:  *
4:  * Computer Science 50
5:  * David J. Malan
6:  *
7:  * Swaps two variables' values.
8:  *
9:  * Demonstrates (clever use of) bitwise operators.
10: *****/
11:
12: #include <stdio.h>
13:
14:
15: // function prototype
16: void swap(int *a, int *b);
17:
18:
19: int
20: main(void)
21: {
22:     int x = 1;
23:     int y = 2;
24:
25:     printf("x is %d\n", x);
26:     printf("y is %d\n", y);
27:     printf("Swapping...\n");
28:     swap(&x, &y);
29:     printf("Swapped!\n");
30:     printf("x is %d\n", x);
31:     printf("y is %d\n", y);
32: }
33:
34:
35: /*
36:  * Swap arguments' values.
37:  */
38:
39: void
40: swap(int *a, int *b)
41: {
42:     *a = *a ^ *b;
43:     *b = *a ^ *b;
44:     *a = *a ^ *b;
45: }
```

## tolower.c

1/1

lectures/7/src/

```
1: /*****
2:  * tolower.c
3:  *
4:  * Computer Science 50
5:  * David J. Malan
6:  *
7:  * Converts an uppercase character to lowercase.
8:  *
9:  * Demonstrates bitwise operators.
10: *****/
11:
12: #include <cs50.h>
13: #include <ctype.h>
14: #include <stdio.h>
15:
16:
17: int
18: main(void)
19: {
20:     // prompt user for an uppercase character
21:     char c;
22:     do
23:     {
24:         printf("Uppercase character please: ");
25:         c = GetChar();
26:     }
27:     while (c < 'A' || c > 'Z');
28:
29:     // print number in lowercase
30:     printf("%c\n", c | 0x20);
31:
32:     // that's all folks
33:     return 0;
34: }
```

**toupper.c**

1/1

lectures/7/src/

```
1: /*****
2:  * toupper.c
3:  *
4:  * Computer Science 50
5:  * David J. Malan
6:  *
7:  * Converts a lowercase character to uppercase.
8:  *
9:  * Demonstrates bitwise operators.
10: *****/
11:
12: #include <cs50.h>
13: #include <ctype.h>
14: #include <stdio.h>
15:
16:
17: int
18: main(void)
19: {
20:     // prompt user for a lowercase character
21:     char c;
22:     do
23:     {
24:         printf("Lowercase character please: ");
25:         c = GetChar();
26:     }
27:     while (c < 'a' || c > 'z');
28:
29:     // print number in lowercase
30:     printf("%c\n", c & 0xdf);
31:
32:     // that's all folks
33:     return 0;
34: }
```

**uint.c**

1/1

lectures/7/src/

```
1: /*****
2:  * uint.c
3:  *
4:  * Computer Science 50
5:  * David J. Malan
6:  *
7:  * Prints a signed 8-bit integer and an unsigned 8-bit integer.
8:  *
9:  * Demonstrates signed, fixed-width types.
10: *****/
11:
12: #include <stdint.h>
13: #include <stdio.h>
14:
15:
16: int
17: main(void)
18: {
19:     // declare and print signed 8-bit integer
20:     int8_t i = 0xff;
21:     printf("%d\n", i);
22:
23:     // declare and print unsigned 8-bit integer
24:     uint8_t u = 0xff;
25:     printf("%d\n", u);
26: }
```