|  |
| --- |
| 1. Modify the repdigit.c program of Section 8.1 so that it shows which digits (if any) were repeated:   1 #include <stdio.h>  2  3 int main()  4 {  5 int check[10] = { 0, };  6 int number;  7  8 printf("Enter a number: ");  9 scanf("%d", &number);  10  11 do {  12 check[number % 10] += 1;  13 number /= 10;  14 } while (number > 0);  15  16 printf("Repeated digit (s): ");  17 for (int i = 0; i < 10; i++) {  18 if (check[i] > 1)  19 printf("%d ", i);  20 } putchar('\n');  21  22 return 0;  23 } |
| 1. Modify the repdigit.c program of Section 8.1 so that it prints a table showing how many times each digit appears in the number:   1 #include <stdio.h>  2  3 int main()  4 {  5 int check[10] = { 0, };  6 int number;  7  8 printf("Enter a number: ");  9 scanf("%d", &number);  10  11 do {  12 check[number % 10] += 1;  13 number /= 10;  14 } while (number > 0);  15  16 printf("%-12s", "Digit:");  17 for (int i = 0; i < 10; i++)  18 printf("%3d", i);  19 putchar('\n');  20  21 printf("%-12s", "Occurrences:");  22 for (int i = 0; i < 10; i++)  23 printf("%3d", check[i]);  24 putchar('\n');  25  26  27 return 0;  28 } |
| 1. Modify the repdigit.c program of Section 8.1 so that the user can enter more than one number to be tested for repeated digits. The program should terminate when the user enters a number that’s less than or equal to 0.   1 #include <stdio.h>  2  3 int main()  4 {  5 int check[9] = { 0, };  6 int number;  7  8 for (;;) {  9 printf("Enter a number: ");  10 scanf("%d", &number);  11  12 if (number <= 0) break;  13  14 do {  15 check[number % 10 - 1] += 1;  16 number /= 10;  17 } while (number > 0);  18 }  19  20 printf("%-12s", "Digit:");  21 for (int i = 0; i < 9; i++)  22 printf("%3d", (i + 1));  23 putchar('\n');  24  25 printf("%-12s", "Occurrences:");  26 for (int i = 0; i < 9; i++)  27 printf("%3d", check[i]);  28 putchar('\n');  29  30  31 return 0;  32 } |
| 1. Modify the reverse.c program of Section 8.1 to use the expression (int)(sizeof(a) / sizeof(a[0])) (or a macro with this value) for the array length.   1 #include <stdio.h>  2  3 #define N 10  4  5 int main()  6 {  7 int a[N], i;  8  9 printf("Enter %d numbers: ", N);  10 for (i = 0; i < (int)(sizeof(a) / sizeof(a[0])); i++)  11 scanf("%d", &a[i]);  12  13 printf("In reverse order:");  14 for (i = (int)(sizeof(a) / sizeof(a[0])) - 1; i >= 0; i--)  15 printf(" %d", a[i]);  16 putchar('\n');  17  18 return 0;  19 } |
| 1. Modify the interest.c program of Section 8.1 so that it compounds interest monthly instead of annually. The form of the output shouldn’t change; the balance should still be shown at annual intervals.   1 #include <stdio.h>  2  3 #define NUM\_RATES ((int) (sizeof(value) / sizeof(value[0])))  4 #define INITIAL\_BALANCE 100.0  5  6 int main(void)  7 {  8 int i, j, low\_rate, num\_years, year;  9 double value[5];  10  11 printf("Enter interest rate: ");  12 scanf("%d", &low\_rate);  13 printf("Enter number of years: ");  14 scanf("%d", &num\_years);  15  16 printf("\nYears");  17 for (i = 0; i < NUM\_RATES; i++) {  18 printf("%6d%%", low\_rate + i);  19 value[i] = INITIAL\_BALANCE;  20 } putchar('\n');  21  22 for (year = 1; year <= num\_years; year++)  23 {  24 printf("%3d ", year);  25 for (i = 0; i < NUM\_RATES; i++) {  26 for (j = 0; j < 12; j++)  27 value [i] += ((low\_rate + i) / 12.0) / 100.0 \* value[i];  28 // value [i] += ((low\_rate + i)) / 100.0 \* value[i];  29 printf("%7.2f", value[i]);  30 } putchar('\n');  31 }  32  33 return 0;  34 } |
| 1. The prototypical Internet newbie is a fellow named B1FF, who has a unique way of writing messages. Here’s typical B1FF communique: Write a “B1FF filter ” that reads a message entered by the user and translates it into B1FF-speak:   1 #include <stdio.h>  2 #include <ctype.h>  3  4 #define MSG\_LEN (100)  5  6 int main(void)  7 {  8 char ch;  9 char message[MSG\_LEN];  10  11 printf("Enter message: ");  12 for (int i = 0; i < MSG\_LEN; i++) {  13 if ( (ch = getchar()) == '\n')  14 break;  15  16 switch ( (ch = toupper(ch)) ) {  17 case 'A': message[i] = '4'; break;  18 case 'B': message[i] = '8'; break;  19 case 'E': message[i] = '3'; break;  20 case 'I': message[i] = '1'; break;  21 case 'O': message[i] = '0'; break;  22 case 'S': message[i] = '5'; break;  23 default: message[i] = ch; break;  24 }  25 }  26  27 printf("In B1FF-speak: %s!!!!!!!!!! \n", message);  28  29 return 0;  30 } |
| 1. Write a program that reads a 5 x 5 array of integers and then prints the row sums and the column sums:   1 #include <stdio.h>  2 #include <ctype.h>  3  4 #define MAT\_SIZ (5)  5  6 int main(void)  7 {  8 int matrix[MAT\_SIZ][MAT\_SIZ];  9 int total\_row[MAT\_SIZ] = { 0, };  10 int total\_col[MAT\_SIZ] = { 0, };  11  12 for (int i = 0; i < MAT\_SIZ; i++) {  13 printf("Enter row %d: ", (i + 1));  14 for (int j = 0; j < MAT\_SIZ; j++) {  15 scanf("%d", &matrix[i][j]);  16 total\_row[i] += matrix[i][j];  17 total\_col[j] += matrix[i][j];  18 }  19 }  20  21 printf("Row totals: ");  22 for (int i = 0; i < MAT\_SIZ; i++)  23 printf("%d ", total\_row[i]);  24 putchar('\n');  25  26 printf("Column totals: ");  27 for (int i = 0; i < MAT\_SIZ; i++)  28 printf("%d ", total\_col[i]);  29 putchar('\n');  30  31 return 0;  32 } |
| 1. Modify Programming Project 7 so that it prompts for five quiz grades for each of five students, then computes the total score and average score for each student, and the average score, high score, and low score for each quiz:   1 #include <stdio.h>  2 #include <ctype.h>  3  4 #define QUIZ\_SIZE (5)  5 #define STUDENT\_SIZE (5)  6  7 int main(void)  8 {  9 int student\_score[STUDENT\_SIZE][QUIZ\_SIZE];  10  11 int student\_total[STUDENT\_SIZE], student\_average[STUDENT\_SIZE],  12 quiz\_high[QUIZ\_SIZE], quiz\_low[QUIZ\_SIZE], quiz\_average[QUIZ\_SIZE];  13  14 for (int i = 0; i < STUDENT\_SIZE; i++) {  15 printf("Student %d: ", (i + 1));  16  17 student\_total[i] = 0;  18 for (int j = 0; j < QUIZ\_SIZE; j++) {  19 scanf("%d", &student\_score[i][j]);  20  21 student\_total[i] += student\_score[i][j];  22  23 if (i == 0)  24 quiz\_average[j] = student\_score[i][j];  25 else  26 quiz\_average[j] += student\_score[i][j];  27 }  28 student\_average[i] = student\_total[i] / QUIZ\_SIZE;  29 }  30  31 for (int i = 0; i < QUIZ\_SIZE; i++) {  32 quiz\_average[i] = quiz\_average[i] / QUIZ\_SIZE;  33 quiz\_low[i] = quiz\_high[i] = student\_score[0][i];  34 for (int j = 1; j < STUDENT\_SIZE; j++) {  35 if (quiz\_high[i] < student\_score[j][i])  36 quiz\_high[i] = student\_score[j][i];  37 if (quiz\_low[i] > student\_score[j][i])  38 quiz\_low[i] = student\_score[j][i];  39 }  40 }  41  42 printf("Student Total Score: ");  43 for (int i = 0; i < STUDENT\_SIZE; i++)  44 printf("%d ", student\_total[i]);  45 putchar('\n');  46  47 printf("Student Average Score: ");  48 for (int i = 0; i < STUDENT\_SIZE; i++)  49 printf("%d ", student\_average[i]);  50 putchar('\n');  51  52 printf("Quiz Average Score: ");  53 for (int i = 0; i < QUIZ\_SIZE; i++)  54 printf("%d ", quiz\_average[i]);  55 putchar('\n');  56  57 printf("Highest Quiz Score: ");  58 for (int i = 0; i < QUIZ\_SIZE; i++)  59 printf("%d ", quiz\_high[i]);  60 putchar('\n');  61  62 printf("Lowest Quiz Score: ");  63 for (int i = 0; i < QUIZ\_SIZE; i++)  64 printf("%d ", quiz\_low[i]);  65 putchar('\n');  66  67  68 return 0;  69 } |
| 1. Write a program that generates a “random walk” across a 10 x 10 array.’   1 #include <stdio.h>  2 #include <stdlib.h>  3 #include <time.h>  4  5 #define MAP\_SIZE\_X (10)  6 #define MAP\_SIZE\_Y (10)  7  8 #define INIT\_POS\_X (0)  9 #define INIT\_POS\_Y (0)  10  11 int main(void)  12 {  13 char map[MAP\_SIZE\_Y][MAP\_SIZE\_X] = { 0, };  14  15 int position\_x, position\_y;  16 int check\_x, check\_y;  17  18 const int dtox[4] = {  19 -1, 0, 1, 0  20 }, dtoy[4] = {  21 0, -1, 0, 1  22 };  23  24 srand((unsigned int)time(NULL));  25  26 position\_x = INIT\_POS\_X, position\_y = INIT\_POS\_Y;  27  28 map[position\_y][position\_x] = 1;  29 for (int step = 1; step < 26; step++) {  30 for (int i = 0, dir = (rand() % 4); i < 4; i++, dir++) {  31 if (dir >= 4) dir = 0;  32  33 check\_x = position\_x + dtox[dir],  34 check\_y = position\_y + dtoy[dir];  35  36 if (check\_x < 0 || check\_y < 0  37 || check\_x >= MAP\_SIZE\_X  38 || check\_y >= MAP\_SIZE\_Y) continue;  39 if (map[ check\_y ][ check\_x ] != 0) continue;  40  41 position\_x = check\_x,  42 position\_y = check\_y;  43  44 map[ position\_y ][ position\_x ] = (step + 1);  45  46 break;  47 }  48  49 if ( map[ position\_y ][position\_x ] == (step - 1) )  50 break;  51 }  52  53 for (int i = 0; i < MAP\_SIZE\_Y; i++) {  54 for (int j = 0; j < MAP\_SIZE\_X; j++) {  55 switch ( map[i][j] ) {  56 case 0: putchar('.'); break;  57 case 1: putchar('A'); break;  58 case 2: putchar('B'); break;  59 case 3: putchar('C'); break;  60 case 4: putchar('D'); break;  61 case 5: putchar('E'); break;  62 case 6: putchar('F'); break;  63 case 7: putchar('G'); break;  64 case 8: putchar('H'); break;  65 case 9: putchar('I'); break;  66 case 10: putchar('J'); break;  67 case 11: putchar('K'); break;  68 case 12: putchar('L'); break;  69 case 13: putchar('M'); break;  70 case 14: putchar('N'); break;  71 case 15: putchar('O'); break;  72 case 16: putchar('P'); break;  73 case 17: putchar('Q'); break;  74 case 18: putchar('R'); break;  75 case 19: putchar('S'); break;  76 case 20: putchar('T'); break;  77 case 21: putchar('U'); break;  78 case 22: putchar('V'); break;  79 case 23: putchar('W'); break;  80 case 24: putchar('X'); break;  81 case 25: putchar('Y'); break;  82 case 26: putchar('Z'); break;  83 } putchar(' ');  84 } putchar('\n');  85 }  86  87 return 0;  88 } |
| 1. Modify Programming Project 8 from Chapter 5 so that the departure times are stored in an array and arrival times are stored in a second array.   1 #include <stdio.h>  2  3 int main(void)  4 {  5 int hour, minute, convert;  6 int departure\_time[8] = {  7 ( 8 \* 60) + 0,  8 ( 9 \* 60) + 43,  9 (11 \* 60) + 19,  10 (12 \* 60) + 47,  11 (14 \* 60) + 0,  12 (15 \* 60) + 45,  13 (19 \* 60) + 0,  14 (21 \* 60) + 45  15 }, arrive\_time[8] = {  16 (10 \* 60) + 16,  17 (11 \* 60) + 52,  18 (13 \* 60) + 31,  19 (15 \* 60) + 0,  20 (16 \* 60) + 8,  21 (17 \* 60) + 55,  22 (21 \* 60) + 20,  23 (23 \* 60) + 58  24 };  25  26 printf("Enter a 24-hour time: ");  27 scanf("%d:%d", &hour, &minute);  28  29 convert = hour \* 60 + minute;  30 hour = minute = -1;  31 for (int i = 0; i < 8; i++) {  32 if (convert < departure\_time[i]) {  33 hour = departure\_time[i] / 60;  34 minute = departure\_time[i] % 60;  35  36 convert = arrive\_time[i];  37 break;  38 }  39 }  40 if (hour == -1 || minute == -1) {  41 printf("You missed all the flight in the day \n");  42 } else {  43 printf("Closest departure time is %d:%.2d %s, ",  44 hour, minute, (hour >= 12) ? ("a.m.") : ("p.m.")  45 );  46  47 printf("arriving at %d:%.2d %s \n",  48 convert / 60, convert % 60, ((convert / 60) >= 12) ? ("a.m.") : ("p.m.")  49 );  50 }  51  52 return 0;  53 } |
| 1. Modify Programming Project 4 from Chapter 7 so that the program labels its output:   1 #include <stdio.h>  2  3 #define PHONE\_MAX (15)  4  5 int main(void)  6 {  7 char phone\_number[PHONE\_MAX];  8  9 printf("Enter phone number: ");  10 for (int i = 0; i < PHONE\_MAX; i++) {  11 phone\_number[i] = getchar();  12 if (phone\_number[i] == '\n')  13 break;  14 }  15  16 printf("In numeric form: ");  17 for (int i = 0; i < PHONE\_MAX; i++) {  18 if (phone\_number[i] == '\n') break;  19  20 switch( phone\_number[i] )  21 {  22 case 'A': case 'B': case 'C':  23 putchar('2');  24 break;  25  26 case 'D': case 'E': case 'F':  27 putchar('3');  28 break;  29  30 case 'G': case 'H': case 'I':  31 putchar('4');  32 break;  33  34 case 'J': case 'K': case 'L':  35 putchar('5');  36 break;  37  38 case 'M': case 'N': case 'O':  39 putchar('6');  40 break;  41  42 case 'P': case 'R': case 'S':  43 putchar('7');  44 break;  45  46 case 'T': case 'U': case 'V':  47 putchar('8');  48 break;  49  50 case 'W': case 'X': case 'Y':  51 putchar('9');  52 break;  53  54 default:  55 putchar( phone\_number[i] );  56 break;  57 }  58 } putchar('\n');  59  60 return 0;  61 } |
| 1. Modify Programming Project 5 from Chapter 7 so that the SCRABBLE values of the letters are stored in an array.   1 #include <stdio.h>  2 #include <ctype.h>  3  4 int main(void)  5 {  6 char value, ch;  7  8 int scrabble\_table[26] = {  9 [0] = 1, [4] = 1, [8] = 1,  10 [11] = 1, [13] = 1, [14] = 1,  11 [17] = 1, [18] = 1, [19] = 1, [20] = 1,  12  13 [3] = 2, [6] = 2,  14  15 [1] = 3, [2] = 3, [12] = 3,  16 [15] = 3,  17  18 [5] = 4, [7] = 4, [21] = 4,  19 [22] = 4, [24] = 4,  20  21 [10] = 5,  22  23 [9] = 8, [23] = 8,  24  25 [16] = 10, [25] = 10,  26 };  27  28 value = 0;  29  30 printf("Enter a word: ");  31 while ( (ch = getchar()) != '\n')  32 {  33 switch ( toupper(ch)) {  34 case 'A': ch = 0; break;  35 case 'B': ch = 1; break;  36 case 'C': ch = 2; break;  37 case 'D': ch = 3; break;  38 case 'E': ch = 4; break;  39 case 'F': ch = 5; break;  40 case 'G': ch = 6; break;  41 case 'H': ch = 7; break;  42 case 'I': ch = 8; break;  43 case 'J': ch = 9; break;  44 case 'K': ch = 10; break;  45 case 'L': ch = 11; break;  46 case 'M': ch = 12; break;  47 case 'N': ch = 13; break;  48 case 'O': ch = 14; break;  49 case 'P': ch = 15; break;  50 case 'Q': ch = 16; break;  51 case 'R': ch = 17; break;  52 case 'S': ch = 18; break;  53 case 'T': ch = 19; break;  54 case 'U': ch = 20; break;  55 case 'V': ch = 21; break;  56 case 'W': ch = 22; break;  57 case 'X': ch = 23; break;  58 case 'Y': ch = 24; break;  59 case 'Z': ch = 25; break;  60 }  61  62 value += scrabble\_table[ch];  63 }  64  65 printf("Scrabble value: %d \n", value);  66  67 return 0;  68 } |
| 1. Modify Programming Project 11 from Chapter 7 so that the program labels its output:   1 #include <stdio.h>  2 #include <ctype.h>  3  4 #define NAME\_LEN (20)  5  6 int main(void)  7 {  8 char ch;  9 signed char first\_name, last\_name[NAME\_LEN];  10  11 int name\_len, check;  12  13 check = first\_name = -1;  14 name\_len = 0;  15  16 printf("Enter a first and last name: ");  17 do {  18 ch = getchar();  19  20 if (ch == '\n')  21 break;  22  23 if (first\_name == -1) {  24 first\_name = ch;  25 continue;  26 }  27  28 if (ch == ' ') {  29 check = 1;  30 continue;  31 }  32  33 if (check > 0)  34 last\_name[name\_len++] = ch;  35 } while (name\_len < NAME\_LEN);  36 last\_name[name\_len] = '\0';  37  38 printf("You entered the name: %c, %s \n", first\_name, last\_name);  39  40 return 0;  41 } |
| 1. Write a program that reverses the words in a sentence:   1 #include <stdio.h>  2 #include <ctype.h>  3  4 #define MAX\_SENTENCE (100)  5  6 int main(void)  7 {  8 char ch, sentence[MAX\_SENTENCE], end\_of\_line;  9 int sentence\_len;  10  11 printf("Enter a sentence: ");  12 for (sentence\_len = 0; sentence\_len < MAX\_SENTENCE; sentence\_len++) {  13 if ( (ch = getchar()) == '\n') break;  14  15 if (ch == '.' || ch == '!' || ch == '?') {  16 end\_of\_line = ch; break;  17 }  18  19 sentence[sentence\_len] = ch;  20 }  21  22 printf("Reversal of sentence: ");  23 for (int end = sentence\_len - 1, start = end; start >= -1; start--) {  24 if (sentence[start] == ' ' || start == -1) {  25 for (int k = (start + 1); k <= end; k++)  26 putchar(sentence[k]);  27  28 putchar((start == -1) ? end\_of\_line : ' ');  29  30 end = start - 1;  31 }  32 } putchar('\n');  33  34 return 0;  35 } |
| 1. Write a program that encrypts a message using a Caesar cipher. The user will enter the message to be encrypted and the shift amount (the number of positions by which letters should be shifted):   1 #include <stdio.h>  2 #include <ctype.h>  3  4 #define MAX\_MSG (100)  5  6 int main(void)  7 {  8 char ch, message[MAX\_MSG];  9 int shift;  10  11 printf("Enter message to be encrypted: ");  12 for (int i = 0; i < MAX\_MSG; i++)  13 if ( (message[i] = getchar()) == '\n')  14 break;  15  16 printf("Enter shift amountt (1-25): ");  17 scanf("%d", &shift);  18  19 printf("Encrypted message: ");  20 for (int i = 0; i < MAX\_MSG; i++)  21 {  22 if ( (ch = message[i]) == '\n') break;  23  24 // Be careful!! This code doesn't have the portability.  25 // It can be only running on the machine with ASCII code page.  26 if (ch >= 'A' && ch <= 'Z')  27 putchar( (((ch - 'A') + shift) % 26 + 'A') );  28 else if (ch >= 'a' && ch <= 'z')  29 putchar( (((ch - 'a') + shift) % 26 + 'a') );  30 else  31 putchar( ch );  32  33 } putchar('\n');  34  35 return 0;  36 } |
| 1. Write a program that tests whether two words are anagrams (permutations of the same letters.)   1 #include <stdio.h>  2 #include <ctype.h>  3  4 #define WORD\_LEN (100)  5  6 int main(void)  7 {  8 int anagram1[26] = { 0, },  9 anagram2[26] = { 0, };  10  11 int position;  12  13 char ch;  14  15 printf("Enter first word: ");  16 for (int i = 0; i < WORD\_LEN; i++) {  17 if ( (ch = getchar()) == '\n' )  18 break;  19  20 switch ( toupper(ch) ) {  21 case 'A': position = 0; break;  22 case 'B': position = 1; break;  23 case 'C': position = 2; break;  24 case 'D': position = 3; break;  25 case 'E': position = 4; break;  26 case 'F': position = 5; break;  27 case 'G': position = 6; break;  28 case 'H': position = 7; break;  29 case 'I': position = 8; break;  30 case 'J': position = 9; break;  31 case 'K': position = 10; break;  32 case 'L': position = 11; break;  33 case 'M': position = 12; break;  34 case 'N': position = 13; break;  35 case 'O': position = 14; break;  36 case 'P': position = 15; break;  37 case 'Q': position = 16; break;  38 case 'R': position = 17; break;  39 case 'S': position = 18; break;  40 case 'T': position = 19; break;  41 case 'U': position = 20; break;  42 case 'V': position = 21; break;  43 case 'W': position = 22; break;  44 case 'X': position = 23; break;  45 case 'Y': position = 24; break;  46 case 'Z': position = 25; break;  47 }  48  49 anagram1[position]++;  50 }  51  52 printf("Enter seconds word: ");  53 for (int i = 0; i < WORD\_LEN; i++) {  54 if ( (ch = getchar()) == '\n' )  55 break;  56  57 switch ( toupper(ch) ) {  58 case 'A': position = 0; break;  59 case 'B': position = 1; break;  60 case 'C': position = 2; break;  61 case 'D': position = 3; break;  62 case 'E': position = 4; break;  63 case 'F': position = 5; break;  64 case 'G': position = 6; break;  65 case 'H': position = 7; break;  66 case 'I': position = 8; break;  67 case 'J': position = 9; break;  68 case 'K': position = 10; break;  69 case 'L': position = 11; break;  70 case 'M': position = 12; break;  71 case 'N': position = 13; break;  72 case 'O': position = 14; break;  73 case 'P': position = 15; break;  74 case 'Q': position = 16; break;  75 case 'R': position = 17; break;  76 case 'S': position = 18; break;  77 case 'T': position = 19; break;  78 case 'U': position = 20; break;  79 case 'V': position = 21; break;  80 case 'W': position = 22; break;  81 case 'X': position = 23; break;  82 case 'Y': position = 24; break;  83 case 'Z': position = 25; break;  84 }  85  86 anagram2[position]++;  87 }  88 /\*  89 printf("anagram1: ");  90 for (int i = 0; i < 26; i++)  91 printf("%3d", anagram1[i]);  92 putchar('\n');  93  94 printf("anagram2: ");  95 for (int i = 0; i < 26; i++)  96 printf("%3d", anagram2[i]);  97 putchar('\n');  98 \*/  99 position = 1;  100 for (int i = 0; i < 26; i++) {  101 if (anagram1[i] != anagram2[i]) {  102 position = -1;  103 break;  104 }  105 }  106  107 printf("The words %s anagrams. \n", (position > 0) ? "are" : "are not");  108  109 return 0;  110 } |
| 1. Write a program that prints an n x n magic square (a square arrangement of the numbers 1, 2, ... n^2 in which the sums of the rows, columns, and diagonals are all the same.)   1 #include <stdio.h>  2  3 #define MAX\_SIZE (100)  4  5 int main(void)  6 {  7 int pos\_x, pos\_y,  8 target\_x, target\_y;  9 int size;  10  11 printf("This program creates a magic square of a specified size. \n");  12 printf("The size must be an odd number between 1 and 99. \n");  13 do {  14 printf("Enter size of magic square: ");  15 scanf("%d", &size);  16 } while ( (size < MAX\_SIZE) && !(size % 2) );  17  18 int map[size][size];  19  20 for (int i = 0; i < size; i++)  21 for (int j = 0; j < size; j++)  22 map[i][j] = 0;  23  24 pos\_x = (size / 2);  25 pos\_y = 0;  26  27 for (int k = 0; k < size \* size; k++) {  28 map[pos\_y][pos\_x] = (k + 1);  29  30 if ( (pos\_y - 1) < 0) target\_y = size - 1;  31 else target\_y = pos\_y - 1;  32 if ( (pos\_x + 1) >= size) target\_x = 0;  33 else target\_x = pos\_x + 1;  34  35 if ( map[target\_y][target\_x] != 0 )  36 pos\_x = pos\_x + 0, pos\_y = pos\_y + 1;  37 else  38 pos\_x = target\_x, pos\_y = target\_y;  39 }  40  41 for (int i = 0; i < size; i++)  42 {  43 for (int j = 0; j < size; j++)  44 printf("%3d", map[i][j]);  45  46 putchar('\n');  47 }  48 return 0;  49 } |