Barak Barclay

Leslie Tekamp

ECE-1021

28 October 2015

HWK-6

5.1 Answer each of the following:

a) A program module in C is called a(n) .

b) A function is invoked with a(n) .

c) A variable that’s known only within the function in which it’s defined is called a(n)

.

d) The statement in a called function is used to pass the value of an expression

back to the calling function.

e) Keyword is used in a function header to indicate that a function does not return

a value or to indicate that a function contains no parameters.

f) The of an identifier is the portion of the program in which the identifier can

be used.

g) The three ways to return control from a called function to a caller are ,

and .

h) A(n) allows the compiler to check the number, types, and order of the arguments

passed to a function.

i) The function is used to produce random numbers.

j) The function is used to set the random number seed to randomize a program.

k) The storage-class specifiers are , , and .

l) Variables declared in a block or in the parameter list of a function are assumed to be of

storage class unless specified otherwise.

m) A non-static variable defined outside any block or function is a(n) variable.

n) For a local variable in a function to retain its value between calls to the function, it must

be declared with the storage-class specifier.

o) The four possible scopes of an identifier are , , and

.

p) A function that calls itself either directly or indirectly is a(n) function.

q) A recursive function typically has two components: one that provides a means for the

recursion to terminate by testing for a(n) case, and one that expresses the

problem as a recursive call for a slightly simpler problem than the original call.

5.2 For the following program, state the scope (either function scope, file scope, block scope or

function-prototype scope) of each of the following elements.

a) The variable x in main.

b) The variable y in cube.

c) The function cube.

d) The function main.

e) The function prototype for cube.

f) The identifier y in the function prototype for cube.

1 #include <stdio.h>

2 int cube( int y );

3

4 int main( void )

5 {

6 int x;

7

8 for ( x = 1; x <= 10; ++x )

9 printf( "%d\n", cube( x ) );

10 }

11

12 int cube( int y )

13 {

14 return y \* y \* y;

15 }

5.3 Write a program that tests whether the examples of the math library function calls shown

in Fig. 5.2 actually produce the indicated results.

5.4 Give the function header for each of the following functions.

a) Function hypotenuse that takes two double-precision floating-point arguments, side1

and side2, and returns a double-precision floating-point result.

b) Function smallest that takes three integers, x, y, z, and returns an integer.

c) Function instructions that does not receive any arguments and does not return a value.

[Note: Such functions are commonly used to display instructions to a user.]

d) Function intToFloat that takes an integer argument, number, and returns a floatingpoint

result.

5.5 Give the function prototype for each of the following:

a) The function described in Exercise 5.4(a).

b) The function described in Exercise 5.4(b).

c) The function described in Exercise 5.4(c).

d) The function described in Exercise 5.4(d).

5.6 Write a declaration for floating-point variable lastVal that’s to retain its value between calls

to the function in which it’s defined.

5.7 Find the error in each of the following program segme nts and explain how the error can be

corrected (see also Exercise 5.46):

a) int g( void )

{

printf( "%s", Inside function g\n" );

int h( void )

{

printf( "%s", Inside function h\n" );

}

}

b) int sum( int x, int y )

{

int result;

result = x + y;

}

c) void f( float a );

{

float a;

printf( "%f", a );

}

d) int sum( int n )

{

if ( 0 == n ) {

return 0; //

}

else {

n + sum( n - 1 );

}

}

e) void product( void )

{

int a, b, c, result;

printf( "%s", "Enter three integers: " )

scanf( "%d%d%d", &a, &b, &c );

result = a \* b \* c;

printf( "Result is %d", result );

return result;

}

Answers to Self-Review Exercises

5.1 a) function. b) function call. c) local variable. d) return. e) void. f) Scope. g) return; or

return expression; or encountering the closing right brace of a function. h) function prototype.

i) rand. j) srand. k) auto, register, extern, static. l) auto. m) external, global. n) static.

o) function scope, file scope, block scope, function-prototype scope. p) recursive. q) base.

5.2 a) Block scope. b) Block Scope. c) File scope. d) File scope. e) File scope. f) Function-prototype

scope.

5.3 See below. [Note: On most Linux systems, you must use the -lm option when compiling

this program.]

1 // ex05\_03.c

2 // Testing the math library functions

3 #include <stdio.h>

4 #include <math.h>

5

6 // function main begins program execution

7 int main( void )

8 {

9 // calculates and outputs the square root

10 printf( "sqrt(%.1f) = %.1f\n", 900.0, sqrt( 900.0 ) );

11 printf( "sqrt(%.1f) = %.1f\n", 9.0, sqrt( 9.0 ) );

12

13 // calculates and outputs the exponential function e to the x

14 printf( "exp(%.1f) = %f\n", 1.0, exp( 1.0 ) );

15 printf( "exp(%.1f) = %f\n", 2.0, exp( 2.0 ) );

16

17 // calculates and outputs the logarithm (base e)

18 printf( "log(%f) = %.1f\n", 2.718282, log( 2.718282 ) );

19 printf( "log(%f) = %.1f\n", 7.389056, log( 7.389056 ) );

20

21 // calculates and outputs the logarithm (base 10)

22 printf( "log10(%.1f) = %.1f\n", 1.0, log10( 1.0 ) );

23 printf( "log10(%.1f) = %.1f\n", 10.0, log10( 10.0 ) );

24 printf( "log10(%.1f) = %.1f\n", 100.0, log10( 100.0 ) );

25

26 // calculates and outputs the absolute value

27 printf( "fabs(%.1f) = %.1f\n", 13.5, fabs( 13.5 ) );

28 printf( "fabs(%.1f) = %.1f\n", 0.0, fabs( 0.0 ) );

29 printf( "fabs(%.1f) = %.1f\n", -13.5, fabs( -13.5 ) );

30

31 // calculates and outputs ceil( x )

32 printf( "ceil(%.1f) = %.1f\n", 9.2, ceil( 9.2 ) );

33 printf( "ceil(%.1f) = %.1f\n", -9.8, ceil( -9.8 ) );

34

35 // calculates and outputs floor( x )

36 printf( "floor(%.1f) = %.1f\n", 9.2, floor( 9.2 ) );

37 printf( "floor(%.1f) = %.1f\n", -9.8, floor( -9.8 ) );

38

39 // calculates and outputs pow( x, y )

40 printf( "pow(%.1f, %.1f) = %.1f\n", 2.0, 7.0, pow( 2.0, 7.0 ) );

41 printf( "pow(%.1f, %.1f) = %.1f\n", 9.0, 0.5, pow( 9.0, 0.5 ) );

42

43 // calculates and outputs fmod( x, y )

44 printf( "fmod(%.3f/%.3f) = %.3f\n", 13.657, 2.333,

45 fmod( 13.657, 2.333 ) );

46

47 // calculates and outputs sin( x )

48 printf( "sin(%.1f) = %.1f\n", 0.0, sin( 0.0 ) );

49

50 // calculates and outputs cos( x )

51 printf( "cos(%.1f) = %.1f\n", 0.0, cos( 0.0 ) );

52

53 // calculates and outputs tan( x )

54 printf( "tan(%.1f) = %.1f\n", 0.0, tan( 0.0 ) );

55 } // end main

sqrt(900.0) = 30.0

sqrt(9.0) = 3.0

exp(1.0) = 2.718282

exp(2.0) = 7.389056

log(2.718282) = 1.0

log(7.389056) = 2.0

log10(1.0) = 0.0

log10(10.0) = 1.0

log10(100.0) = 2.0

fabs(13.5) = 13.5

fabs(0.0) = 0.0

fabs(-13.5) = 13.5

ceil(9.2) = 10.0

ceil(-9.8) = -9.0

floor(9.2) = 9.0

floor(-9.8) = -10.0

pow(2.0, 7.0) = 128.0

pow(9.0, 0.5) = 3.0

fmod(13.657/2.333) = 2.010

sin(0.0) = 0.0

cos(0.0) = 1.0

tan(0.0) = 0.0

5.4 a) double hypotenuse( double side1, double side2 )

b) int smallest( int x, int y, int z )

c) void instructions( void )

d) float intToFloat( int number )

5.5 a) double hypotenuse( double side1, double side2 );

b) int smallest( int x, int y, int z );

c) void instructions( void );

d) float intToFloat( int number );

5.6 static float lastVal;

5.7 a) Error: Function h is defined in function g.

Correction: Move the definition of h out of the definition of g.

b) Error: The body of the function is supposed to return an integer, but does not.

Correction: Delete variable result and place the following statement in the function:

return x + y;

c) Error: Semicolon after the right parenthesis that encloses the parameter list, and redefining

the parameter a in the function definition.

Correction: Delete the semicolon after the right parenthesis of the parameter list, and

delete the declaration float a; in the function body.

d) Error: The result of n + sum( n - 1 ) is not returned; sum returns an improper result.

Correction: Rewrite the statement in the else clause as

return n + sum( n - 1 );

e) Error: The function returns a value when it’s not supposed to.

Correction: Eliminate the return statement.

//

// HWK6(2).cpp

//

// By: Barak Barclay

// Date: 28 Oct 2015

//

// Problem Statement: Write a program with a user defined function. The main function should prompt the user to enter an integer number, than call the user function. Pass the entered number to the user function which will return a Boolean. The user function will implement a sentinel loop, prompting the user to enter another integer number, than check to see if that number is a divisor (remainder = 0) of the number passed to the user function. Return true if the number is a divisor, remain in the sentinel loop until an entered number is a divisor.

//

// %%%% Algorthim %%%%

//

// preprocessor directives

// init uf

// start main

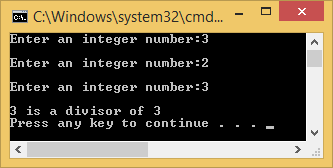
// init variable

// prompt user to enter integer

// read in n

// call uf

// end main

//

// 1111 uf 1111

// start uf

// init variable

// prompt user to enter integer

// read in n2

// while

// prompt user to enter int

// read in int

// end while

// print result

// return true

// end uf

//

#include <stdio.h>

#pragma warning(disable : 4996)

bool uf(int Fun\_n);

void main(void)

{

int n = 0;

printf("Enter an integer number:");

scanf("%d", &n);

printf("\n");

uf(n);

}

bool uf(int Fun\_n)

{

int n2 = 0;

printf("Enter an integer number:");

scanf("%d", &n2);

printf("\n");

while (Fun\_n%n2 != 0)

{

printf("Enter an integer number:");

scanf("%d", &n2);

printf("\n");

}

printf("%d is a divisor of %d\n", n2, Fun\_n);

return true;

}

//

// HWK6(3).cpp

//

// By: Barak Barclay

// Date: 28 Oct 2015

//

// Problem Statement: Write a program with three user defined functions. The main function should prompt the user to enter two float numbers corresponding to the sides of a rectangle. Call the first user function, pass the two rectangle sides, and return a double corresponding to the perimeter of the rectangle. Print the perimeter to the screen. Call the second user function, pass the two rectangle sides, and return a double corresponding to the diameter of the rectangle. Print the diameter to the screen. Call the third user function, pass the two rectangle sides, and return a double corresponding to the area of the rectangle. Print the area to the screen.

//

// %%%% Algorthim %%%%

//

// preprocessor directives

// init uf’s

// start main

// init variables

// enter and read in variables

// print what uf’s return

// end main

//

// 1111 uf1 1111

// start uf1

// init variables

// calculations

// return

// end uf1

// 2222 uf2 2222

// start uf2

// init variables

// calculations

// return

// end uf2

// 3333 uf3 3333

// start uf3

// init variables

// calculations

// return

// end uf3

//

#include <stdio.h>

#include <math.h>

#pragma warning(disable : 4996)

double uf1(float Fun\_f1, float Fun\_f2);

double uf2(float Fun\_f1, float Fun\_f2);

double uf3(float Fun\_f1, float Fun\_f2);

void main(void)

{

float f1 = 0.0;

float f2 = 0.0;

printf("Enter a float number corresponding to a side of a rectangle:");

scanf("%f", &f1);

printf("\n");

printf("Enter a float number corresponding to the other side of the rectangle:");

scanf("%f", &f2);

printf("\n");

printf("Parameter is: %lf", uf1(f1, f2));

printf("\n");

printf("Diameter is: %lf", uf2(f1, f2));

printf("\n");

printf("Area is: %lf", uf3(f1, f2));

printf("\n\n");

}

double uf1(float Fun\_f1, float Fun\_f2)

{

double p = 0.0;

p = 2 \*(double)Fun\_f1 + 2 \* (double)Fun\_f2;

return p;

}

double uf2(float Fun\_f1, float Fun\_f2)

{

double c = 0.0;

double d = 0.0;

c = pow((double)Fun\_f1, 2.0) + pow((double)Fun\_f2, 2.0);

d = sqrt(c);

return d;

}

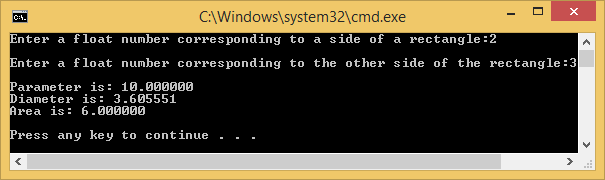
double uf3(float Fun\_f1, float Fun\_f2)

{

double a = (double)Fun\_f1 \* (double)Fun\_f2;

return a;

}



//

// HWK6(4).cpp

//

// By: Barak Barclay

// Date: 28 Oct 2015

//

// Problem Statement: Write a program with one user function which generates a multiplication table using integer numbers from 0 to n. In the main function, prompt the user to enter in an integer number corresponding to the size of the multiplication table. Read in the number. Call the user function. Pass the entered number, return nothing. The user function will generate and prompt to the screen a square multiplication from zero to the entered number (n).

//

// %%%% Algorthim %%%%

//

// preprocessor directives

// init uf

// start main

// prompt user to enter variable

// read in variable

// call uf

// end main

//

// 1111 uf1 1111

// start uf1

// init variables

// for loop

// for loop

// print variables

// end for

// print new line

// end for

// end uf1

#include <stdio.h>

#include <math.h>

#pragma warning(disable : 4996)

void uf(int Fun\_n);

void main(void)

{

int n = 0;

printf("Enter a integer number corresponding to the size of a multiplication table:");

scanf("%d", &n);

printf("\n");

uf(n);

}

void uf(int Fun\_n)

{

int i;

int j;

for (j = 1; j <= Fun\_n; j++)

{

for (i = 1; i <= Fun\_n; i++)

{

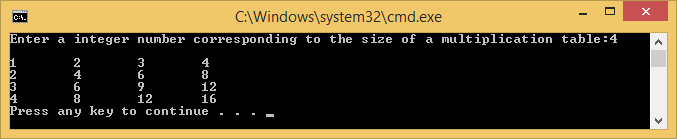
printf("%d\t", j\*i);

}

printf("\n");

}

}



//

// HWK6(5).cpp

//

// By: Barak Barclay

// Date: 28 Oct 2015

//

// Problem Statement: Modify the function created in Exercise 5.19 to form the square out of whatever character is contained in character parameter fillCharacter. Thus if side is 5 and fillCharacter is “#”, then this function should print:

#####

#####

#####

#####

#####

// %%%% Algorthim %%%%

//

// preprocessor directives

// start main

// init variables

// enter and read in variables

// for

// for

// print variable

// end for

// print new line

// end for

// end main

//

#include <stdio.h>

#pragma warning(disable : 4996)

void main(void)

{

int n = 0;

char a = 'a';

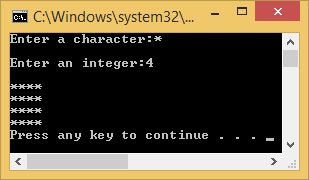
int i;

int j;

printf("Enter a character:");

scanf("%c", &a);

printf("\n", a);

 printf("Enter an integer:");

scanf("%d", &n);

printf("\n", a);

for (i = 0; i < n; i++)

{

for (j = 0; j < n; j++)

{

printf("%c", a);

}

printf("\n");

}

}

//

// HWK6(6).cpp

//

// By: Barak Barclay

// Date: 28 Oct 2015

//

// Problem Statement: Write a recursive function power( base, exponent ) that when invoked returns baseexponent For example, power( 3, 4 ) = 3 \* 3 \* 3 \* 3. Assume that exponent is an integer greater than or equal to 1. Hint: The recursion step would use the relationship baseexponent = base \* baseexponent–1 and the terminating condition occurs when exponent is equal to 1 because base1 = base

//

// %%%% Algorthim %%%%

//

// preprocessor directives

// init uf

// start main

// init variables

// enter and read in variables

// print what uf returns

// end main

//

// 1111 uf1 1111

// start uf

// if

// return Fun\_b

// end if

// else

// return eqn

// end else

// end uf

//

#include <stdio.h>

#include <math.h>

#pragma warning(disable : 4996)

double uf(double Fun\_b, double Fun\_p);

void main(void)

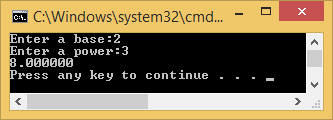
{

double b = 2.0;

double p = 3.0;

printf("Enter a base:");

scanf("%lf", &b);

 printf("Enter a power:");

scanf("%lf", &p);

printf("%lf", uf(b, p));

printf("\n");

}

double uf(double Fun\_b, double Fun\_p)

{

if ((int)Fun\_p == 1)

{

return Fun\_b;

}

else

{

return (Fun\_b \* pow(Fun\_b, (Fun\_p - 1)));

}

}

//

// HWK6(7).cpp

//

// By: Barak Barclay

// Date: 28 Oct 2015

//

// Problem Statement: Write a program to print out the values of a sine function – see below.

x[i] = Amp \* cos( freq \* PI \* i / 20)

The main function should generate a random number between 1 and 10 (inclusive). Use this for the Amp value in the given equation. Print the Amp value to the screen. The main function should then generate a random float number between 10 and 50. The formula below will generate a random float between zero and 0.99999 (i.e. – up to but not including one).

(float) rand() / (float) (RAND\_MAX + 1);

use this for the freq value in the first equation. Print the freq value to the screen. Call a user defined function, pass it the two random numbers, calculate 51 values ( i=0 to i=50) of the sine function and print them to the screen. Return nothing.

//

// %%%% Algorthim %%%%

//

// preprocessor directives

// init uf

// start main

// init variables

// calculations

// print variables

// call uf

// end main

//

// 1111 uf1 1111

// start uf

// init variables

// for

// calculation

// print variables

// end for

// end uf

//

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#pragma warning(disable : 4996)

void uf(int Fun\_amp, float Fun\_freq);

void main(void)

{

int amp = 0;

float randnum = 0;

float freq = 0;

amp = (rand() % 10) + 1;

printf("Amp is: %d \n", amp);

randnum = (float)(rand() % 40) + (float)10.0;

freq = (float)randnum / (float)(RAND\_MAX + 1);

printf("Freq is: %d \n", freq);

uf(amp, freq);

}

void uf(int Fun\_amp, float Fun\_freq)

{

float x = 0.0;

double Pi = 3.14159265359;

int i;

for (i = 0; i <= 50; i++)

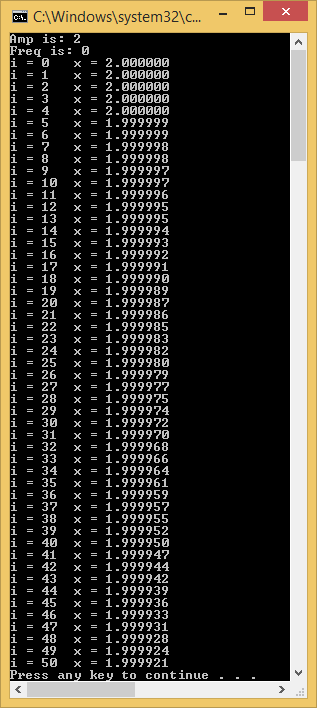
{

x = (float)Fun\_amp \* (float)cos(Fun\_freq \* Pi \* (float)i / 20.0);

printf("i = %d\tx = %f\n", i, x);

}

}



//

// HWK6(8).cpp

//

// By: Barak Barclay

// Date: 28 Oct 2015

//

// Problem Statement: Write a program that will randomly pick a card from a deck of 52 cards – assume the card is replaced into the deck after each pick. There are four possible suits, 13 possible cards. Use a sentinel loop, ‘y’ to continue ‘n’ to quit. Write a function to randomly pick the suit. Write another function to randomly pick the card value (1=ace to 13=King). The master user function will call the other two user functions for suit and value. And print the card (suit and value) to the screen. The main will write a message to the screen, asking the user if they want a new card. If ‘y’ call the master user function. If ‘n’ terminate the program.

//

// %%%% Algorthim %%%%

//

// preprocessor directives

// init uf’s

// start main

// init variables

// while

// flush

// print prompt

// read in variable

// if

// break

// end if

// call master

// end main

//

// 1111 master 1111

// start uf

// print suit and value

// end uf

//

// 2222 suit 2222

// start uf

// calculate and return suit

// end uf

//

// 3333 value 3333

// start uf

// calculate and return value

// end uf

//

#include <stdio.h>

#include <stdlib.h>

#pragma warning(disable : 4996)

void master();

int suit();

int value();

void main(void)

{

char a = 'y';

while (a != 'n')

{

fflush(stdin);

printf("Would you like a card, yes(y) or no(n)?:");

scanf("%c", &a);

if (a == 'n')

{

break;

}

master();

}

}

void master()

{

int suit1 = suit();

switch (suit1)

{

case 1:

{

printf("Value:%d\tSuit:Diamonds\n", value());

break;

} //end case

case 2:

{

printf("Value:%d\tSuit:Hearts\n", value());

break;

} //end case

case 3:

{

printf("Value:%d\tSuit:Clubs\n", value());

break;

} //end case

case 4:

{

printf("Value:%d\tSuit:Spades\n", value());

break;

} //end case

default:

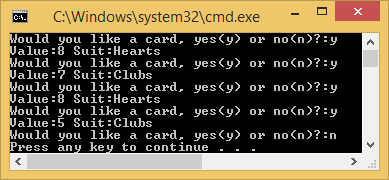
{

printf("\n\n ERROR \n\n");

break;

} //end case

} //end switch

}

int suit()

{

int suit = 0;

suit = (rand() % 4) + 1;

return suit;

}

int value()

{

int value = 0;

value = (rand() % 13) + 1;

return value;

}

//

// HWK6(9).cpp

//

// By: Barak Barclay

// Date: 28 Oct 2015

//

// Problem Statement: Use a single-subscripted array to solve the following problem. Read in 20 numbers, each of which is between 10 and 100, inclusive. As each number is read, print it only if it’s not a duplicate of a number already read. Provide for the “worst case” in which all 20 numbers are different. Use the smallest possible array to solve this problem.

//

// %%%% Algorthim %%%%

//

// preprocessor directives

// start main

// init variable

// print prompt

// for

// set d = 0

// read in variable

// while

// error and read in

// end while

// for

// if

// set d = 1

// break

// end if

// end for

// if

// set array

// end if

// end for

// print result

// end main

//

#include <stdio.h>

#define SIZE 20

#pragma warning(disable : 4996)

void main(void)

{

int list[SIZE] = { 0 };

int i;

int j;

int c = 0;

int d;

int n;

printf("Enter 20 integers between 10 and 100:\n");

for (i = 0; i < SIZE; i++)

{

d = 0;

scanf("%d", &n);

while (n < 10 || n > 100)

{

printf("\n\n ERROR - Invalid Input \n\n");

scanf("%d", &n);

}

for (j = 0; j < c; j++)

{

if (n == list[j])

{

d = 1;

break;

}

}

if (!d)

{

list[c++] = n;

}

}

printf("\nNon-duplcate integers:\n");

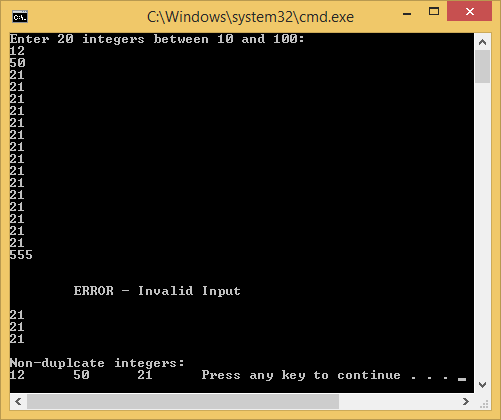
for (i = 0; i < SIZE && list[i] != 0; i++)

{

printf("%d\t", list[i]);

}

}



10)

//

// HWK6(10).cpp

//

// By: Barak Barclay

// Date: 28 Oct 2015

//

// Problem Statement: Use a single-subscripted array to solve the following problem. Read in 20 numbers, each of which is between 10 and 100, inclusive. As each number is read, print it only if it’s not a duplicate of a number already read. Provide for the “worst case” in which all 20 numbers are different. Use the smallest possible array to solve this problem.

//

// %%%% Algorthim %%%%

//

// preprocessor directives

// start main

// init variable

// do while loop (num >= 0)

// switch statement

// end while

// end main

//

#include <stdio.h>

#define SIZE 80

#pragma warning(disable : 4996)

void uf1(float \*a, size\_t \*i);

void uf2(float Fun\_a, size\_t Fun\_i);

void main(void)

{

float a[SIZE];

size\_t i;

uf1(a[SIZE], &i);

uf2(a[SIZE], i);

}

void uf1(float \*a, size\_t \*i)

{

for (\*i = 0; \*i < SIZE; \*i++)

{

a[\*i]\* = 10;

}

}