Michael Campbell

X52.9008 – 01

Tues 6:30-9:00PM

June 22, 2010

HW # 5 --Chapter 8 # 4, 7, 11

Chapter 13 # 3, 4, 5, 9

Appendix B

8.4) Modify Program 8.8 so that the value of guess is printed each time through the while loop. Notice how quickly the value of guess converges to the square root. What conclusions can you reach about the number of iterations through the loop, the number whose square root is being calculated, and the value of the initial guess?

// Function to calculate the absolute value of a number

#include <stdio.h>

float absoluteValue (float x)

{

if (x < 0)

x = -x;

return (x);

}

//Function to compute the square root of a number

float squareRoot (float x)

{

const float epsilon = .00001;

float guess = 1.0;

while ( absoluteValue (guess \* guess - x) >= epsilon ){

guess = ( x / guess + guess ) / 2.0;

printf("guess = %f\n", guess);

}

return guess;

}

int main (void)

{

printf ("squareRoot (2.0) = \t%f\n", squareRoot (2.0));

printf ("squareRoot (144.0) = \t%f\n", squareRoot (144.0));

printf ("squareRoot (17.5) = \t%f\n", squareRoot (17.5));

return 0;

}

Output:

guess = 1.500000

guess = 1.416667

guess = 1.414216

squareRoot (2.0) = 1.414216

guess = 72.500000

guess = 37.243103

guess = 20.554796

guess = 13.780231

guess = 12.114992

guess = 12.000546

guess = 12.000000

squareRoot (144.0) = 12.000000

guess = 9.250000

guess = 5.570946

guess = 4.356122

guess = 4.186728

guess = 4.183301

guess = 4.183300

squareRoot (17.5) = 4.183300

The bigger the value to be evaluated, the more number of iterations are required to find the square root.

8.7) Write a function that raises an integer to a positive integer power. Call the function x\_to\_the\_n taking two integer arguments x and n. Have the function return a long int, which represents the results of calculating xn.

#include <stdio.h>

long int x\_to\_the\_n (int x, int n)

{

int i;

int result;

result = 1;

for (i = 1; i <= n && n > 0; i++)

result \*= x;

return result;

}

int main (void)

{

int x = 4;

int n = 3;

long int x\_to\_the\_n (int x, int n);

printf("\nResult = %li\n\n", x\_to\_the\_n(x, n));

return 0;

}

Output:

Result = 64

8.11) Write a function called arraySum that takes two arguments: an integer array and the number of elements in the array. Have the function return as its result the sum of the elements in the array.

#include <stdio.h>

long int arraySum (int array[], int num)

{

int i;

int result;

result = 0;

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

result += array[i];

return result;

}

int main (void)

{

int x[11] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

long int arraySum (int array[], int num);

printf("\nResult = %li\n\n", arraySum(x, 11));

return 0;

}

Output:

Sum = 55

13.3) Define a macro MIN that gives the minimum of two values. Then write a program to test the macro definition.

#include <stdio.h>

#define MIN(a,b) (((a) < (b)) ? (a) : (b))

int main (void)

{

int x = 733;

int y = 101;

printf("Min = %i\n", MIN(x,y));

return 0;

}

Output:

Min = 101

13.4) Define a macro MAX3 that gives the maximum of three values. Write a program to test the definition.

#include <stdio.h>

#define MAX3(a,b,c) (a > b) ? ((a > c) ? a : c) : ((b > c) ? b : c);

int main (void)

{

int x = 7;

int y = 101;

int z = 44;

int maximum;

maximum = MAX3(x,y,z);

printf("Max = %i\n", maximum);

return 0;

}

Output:

Max = 101

13.5) Write a macro SHIFT to perform the identical purposes as the shift function of Program 12.3

#include <stdio.h>

#define SHIFT(x, y) (((y) > 0) ? (x <<= y) : (x >>= -y))

int main (void)

{

unsigned int w1 = 0177777u, w2 = 0444u;

printf("%o\t%o\n", SHIFT(w1, 5), w1 << 5);

printf("%o\t%o\n", SHIFT(w1, -6), w1 >> 6);

printf("%o\t%o\n", SHIFT(w2, 0), w2 >> 0);

return 0;

}

Output:

7777740 7777740

77777 77777

444 444

13.9) Write a macro ABSOLUTE\_VALUE that computes the absolute value of its argument. Make certain that an expression such as

ABSOLUTE\_VALUE (x + delta)

Is properly evaluated by the macro.

#include <stdio.h>

#define ABSOLUTE\_VALUE(x) (((x) < 0) ? -(x) : (x))

int main (void) {

int a = -1;

int b = 1;

int c = 0;

printf("Absolute value of %i is %i\n", a, ABSOLUTE\_VALUE(a));

printf("Absolute value of %i is %i\n", b, ABSOLUTE\_VALUE(b));

printf("Absolute value of %i is %i\n", c, ABSOLUTE\_VALUE(c));

return 0;

}

Output:

Absolute value of -1 is 1

Absolute value of 1 is 1

Absolute value of 0 is 0