САА - Lab 3

Problem 21\*

Design an algorithm that counts the number of pairs of symmetric (to the main diagonal) elements for which the value of the element above the main diagonal is less than the value of the element below the main diagonal. For example, the following pairs of elements are symmetric about the main diagonal: а10 and а01; а20 and а02; а21 and а12; а30 and а03 and so on.

а00 а01 а02 а03

а10 а11 а12 а13

а20 а21 а22 а23

а30 а31 а32 а33

VII.Recursion

Problem 22

Calculating the factorial of a number using recursion.

#include <iostream>

using namespace std;

int fact(int n)

{

if (n <= 1)

return 1;

else

return n\*fact(n-1);

}

int main()

{

int n;

cout << "n = ";

cin >> n;

cout << "n! = " << fact(n) << "\n";

return 0;

}

Problem 23

Calculating the factorial of a number using a loop.

#include <iostream>

using namespace std;

int fact(int n)

{

if(n == 0)

return 1;

int i, f = 1;

for(i=1; i<=n; i++)

f = f\*i;

return f;

}

int main()

{

int n;

cout << "n = ";

cin >> n;

cout << "n! = " << fact(n) << "\n";

return 0;

}

Problem 24\*

Design an algorithm that converts a positive decimal integer to binary using a recursive function.

Problem 25\*

Design an algorithm that calculates xn (n is an integer) using a recursive function and the following formulas:

xn = x\*xn-1, n>0

xn = 1, n=0

xn = 1/(x\*xn-1), n<0.

Problem 26

Using a recursive function to find the greatest common divisor of two natural numbers.

#include <iostream>

using namespace std;

int min(int p, int q)

{

if (p > q)

return q;

else

return p;

}

int gcd(int c, int d)

{

if( c == d)

return c;

else

return gcd(abs(c-d), min(c,d));

}

int main()

{

int a, b;

cout << "a = ";

cin >> a;

cout << "b = ";

cin >> b;

cout << "GCD = " << gcd(a,b) << "\n";

return 0;

}

Problem 27\*

Design an algorithm that calculates the sum of the elements of a given one-dimensional array using a recursive function.

Problem 28\*

Design an algorithm that checks for the presence of a given digit in a natural number using a recursive function.

Problem 29

Outputting the digits of a given natural number in reverse order using a recursive function.

#include <iostream>

using namespace std;

void reverse(int n)

{

if(n < 10)

cout << n;

else

{

cout << n%10;

reverse(n/10);

}

}

int main()

{

int n;

cout << "n: ";

cin >> n;

reverse(n);

cout << endl;

return 0;

}

Problem 30

Using a recursive function to check for the presence of a given value inside a one-dimensional array.

#include <iostream>

using namespace std;

#define N 5

int a[N];

int search(int x, int n)

{

if(n<1)

return a[0]==x;

return x==a[n-1] || search(x,n-1);

}

int main()

{

int i, n;

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

{

cout << "array[" << i << "] = ";

cin >> a[i];

}

cout << "n: ";

cin >> n;

cout << "n = " << n;

if(search(n,i))

cout << " is ";

else

cout << " is not ";

cout << "an element of the array" << endl;

return 0;

}

Problem 31

A program that uses a recursive function to check whether a given sequence is monotonically decreasing, i.e. whether each element of the array is greater than or equal to the next.

#include <iostream>

using namespace std;

#define N 5

int a[N];

bool monotone(int n)

{

bool b;

if(n==1)

return 1;

if(a[n-2]>=a[n-1])

b=1;

else b=0;

return b && monotone(n-1);

}

int main()

{

int i;

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

{

cout << "array[" << i << "] = ";

cin >> a[i];

}

cout<<"The sequence ";

if(monotone(i))

cout<<"is ";

else

cout<<"is not ";

cout << "monotonic. " << endl;

return 0;

}

1. Run and analyze the solved problems – 22, 23, 26, 29, 30 and 31.

2. Design algorithms and write programs that solve the remaining problems – 21\*, 24\*, 25\*, 27\* and 28\*.