

# DATA STRUCTURES AND ALGORITHMS IN C++ (Tom 1)

*Thân mến tặng các sinh viên  
Học viện Bưu chính Viễn Thông*

*Ngày 10 Tháng 8 Năm 2020*

*Tác giả*



*Nguyễn Xuân Huy*

# CONTENTS

Schedule	3
Hello	4
Equations	5
Proper factors	7
Roman Numerals	10
Roman Fibonacci	17
Expression	23
Tom and Jerry	43
Sieve of Eratosthenes	48
10001st prime (Problem 7 Project Euler)	52
Largest prime factor (Problem 3 Project Euler)	55
Queens	58
Rhythm	63
Range Check Error	67
New RomanNumerals	69
New Roman Fibonacci (R+)	73
New Roman Fibonacci Again (R++)	77

## Schedule

4 lớp, mỗi lớp 2 buổi / tuần. Thời gian là 4 tuần

Lớp 1: sáng thứ 2, sáng thứ 4.

Lớp 2: chiều thứ 2, chiều thứ 4.

Lớp 3: sáng thứ 3, sáng thứ 5.

Lớp 4: chiều thứ 3, chiều thứ 5.

Phòng tầng 1, Samsung Lab.

		Mon	Tue	Wed	Thu
week 1	Morning	I	III	I	III
Aug 10 2020	Afternoon	II	IV	II	IV
week 2	Morning	I	III	I	III
Aug 17 2020	Afternoon	II	IV	II	IV
week 3	Morning	I	III	I	III
Aug 24 2020	Afternoon	II	IV	II	IV
week 4	Morning	I	III	I	III
Aug 31 2020	Afternoon	II	IV	II	IV

## Hello

Con của bạn = chương trình của bạn

Nuôi dưỡng con = phát triển chương trình

Đầu tiên phải sinh một đứa con

```
/* *****  
 * hello  
 *  
 *  
 * ***** */  
#include <iostream>  
using namespace std;  
main() {  
    cout << "\n Main says: Hello World!";  
    return 0;  
} // main
```

## Equations

### Clean Room (IBM)

```

/*****
 * Equation of degree 2
 * ax^2 + bx + c = 0
 *
 *****/

#include <iostream>
#include <cmath>

using namespace std;

// ax^2 + bx + c = 0
int Equation2(double a, double b, double c, double &x1, double &x2) {
    double delta = b*b - 4*a*c;
    if (delta < 0) {
        return 0; // no solution
    }
    if (delta == 0) {
        x1 = x2 = -b / (2*a);
        return 1; // double solution
    }
    // delta > 0
    double Vdelta = sqrt(delta);
    x1 = (-b + Vdelta)/(2*a);
    x2 = -b/a - x1;
    return 2;
} // Equation2

void Test () {
    // 2, 3, 5
    // (x-2)(x-3) = x^2 - 5x + 6: 1, -5, 6
    // (x-3)^2 = x^2 - 6x + 9: 1, -6, 9
    double x1, x2;
    switch(Equation2(1, -6, 9, x1, x2)) {
        case 0: cout << "\n No solution.";
                break;
        case 1: cout << "\n x1 = x2 = " << x1;
                break;
        case 2: cout << "\n x1 = " << x1 << "    x2 = " << x2;
                break;
    } // switch
} // Test

main() {
```

```
Test();  
return 0;  
} //
```

## Proper factors

Nếu số nguyên dương  $x$  chia hết cho  $u$  thì  $u$  được gọi là *ước số của  $x$* .

Nếu  $u < x$  thì  $u$  được gọi là *ước thực sự của  $x$* .

Ví dụ, 12 có 6 ước là 1, 2, 3, 4, 6 và 12, trong đó 5 số đầu tiên là những ước thực sự.

Tìm số lượng các ước thực sự của các số dưới 1M.

Improve, improve and improve again

### Factor Version 1: Brute force

```
/******  
 * FACTOR.CPP (proper factors)  
 * Version 1. Brute Force (no luc thun tuy / vet can)  
 *****/  
  
#include <iostream>  
#include <cmath>  
  
using namespace std;  
  
const int MN = 1000000;  
  
int Tau(int x) {  
    int count = 0;  
    for (int a = 1; a < x; ++a)  
        if (x % a == 0)  
            ++count;  
    return count;  
} // Tau  
  
int Total(int n) {  
    int s = 0;  
    for (int x = 1; x < n; ++x)  
        s += Tau(x);  
    return s;  
} // Total  
  
main() {  
    cout << Total(100); // 374  
    cout << Total(MN); // ???  
    cout << " T H E      E N D";  
    return 0;  
} // main
```

## Factor Version 2: Improve function Tau

```

/*****
 * FACTOR.CPP (proper factors)
 * Version 2. Improve Tau
 *****/

#include <iostream>
#include <cmath>
#include <ctime>

using namespace std;

const int MN = 1000000;

int Tau(int x) {
    int count = 1;
    int sx = int(sqrt(x));
    for (int a = 2; a <= sx; ++a) // 1 2 3 6
        if (x % a == 0)
            count += 2;
    return (sx*sx == x) ? count-1 : count;
} // Tau

int Total(int n) {
    int s = 0;
    for (int x = 1; x < n; ++x)
        s += Tau(x);
    return s;
} // Total

main() {
    int t1 = time(NULL);
    cout << "\n " << Total(MN); // 12969986
    cout << "\n Time = " << difftime(time(NULL),t1); // 2s
    cout << "\n T H E      E N D";
    return 0;
} // main

```

## Factor Version 3: Compute all Tau

```

/*****
 * FACTOR.CPP (proper factors)
 * Version 3. Compute all Tau
 *****/

#include <iostream>
#include <cmath>
#include <ctime>

using namespace std;

const int MN = 1000000;

```



```

int t[MN];

// t[i] = Tau(i), 1 <= i < n
int AllTau(int n) {
    for (int i = 1; i < n; ++i) t[i] = 0;
    int n2 = n / 2;
    for (int i = 1; i <= n2; ++i)
        for (int j = i+i; j < n; j += i)
            ++t[j];
} // Tau

int Total(int n) {
    AllTau(n);
    int s = 0;
    for (int x = 1; x < n; ++x)
        s += t[x];
    return s;
} // Total

main() {
    int t1 = time(NULL);
    cout << "\n " << Total(MN); // 12969986
    cout << "\n Time = " << difftime(time(NULL),t1); // 0s
    cout << "\n T H E      E N D";
    return 0;
} // main

```

## Roman Numerals

5E Hãy khảo sát cẩn thận chỉ ít 5 ví dụ

### Roman Numerals Version 1: Roman To Integer

```
/******
Roman Numerals (Ver. 1)
ToInt: Roman -> Integer

*****/
#include <iostream>
#include <cmath>
using namespace std;

int ival[] =
{1000, 900, 500, 400, 100, 90, 50, 40, 10, 9, 5, 4, 1};

string rval[] =
{"M", "CM", "D", "CD", "C", "XC", "L", "XL", "X", "IX", "V", "IV", "I"};
int len = 13;

void Go() {
    cout << " ? ";
    fflush(stdin);
    if (cin.get() == '.')
        exit(0);
}

// Vị trí xuất hiện kí tự c trong string s
int Cpos(char c, string s) {
    for (int i = 0; i < s.length(); ++i)
        if (s[i] == c) return i;
    return -1;
} // Cpos

// Vị trí xuất hiện string r trong rval
int Rpos(string r) {
    for (int i = 0; i < len; ++i)
        if (rval[i] == r) return i;
    return -1;
} // Rpos

// Vị trí xuất hiện int v trong ival
int Ipos(int v) {
    for (int i = 0; i < len; ++i)
        if (ival[i] == v) return i;
    return -1;
} // Ipos

// rom r -> int
```

```

int RomInt(string r) {
    int i = Rpos(r);
    return (i < 0) ? 0 : ival[i];
} // RomInt

// int v -> rom
string IntRom(int v) {
    int i = Ipos(v);
    return (i < 0) ? "" : rval[i];
} // IntRom

string Str(char c) {
    string s = "";
    s += c;
    return s;
} // Str

int ToInt(string inpr) { // convert rom to int CDLVI -> 456
    // Bo cac dau cach
    string r = "";
    for (int i = 0; i < inpr.length(); ++i)
        if (inpr[i] != 32) r += inpr[i];

    if (r == "") return 0;

    r += "#"; // Them linh canh cuoi xau: guards
    int n = r.length();
    int val = 0; // output value
    int i = 0;
    string s;

    while (i < n) {
        char c = r[i]; // ki tu hien hanh
        i++; // ki tu sat sau c

        if (Cpos(c, "MDLV") >= 0) {
            val += RomInt(Str(c));
            continue;
        }

        if (c == 'C') { // Xet them ki tu r[i] sat sau c: CM, CD
            s = Str(c);
            if (Cpos(r[i], "MD") >= 0) {
                s += r[i];
                i++;
            }
            val += RomInt(s);
            continue;
        }

        if (c == 'X') { // Xet them ki tu r[i] sat sau c: XL, XC
            s = Str(c);
            if (Cpos(r[i], "LC") >= 0) {
                s += r[i];
                ++i;
            }
        }
        val += RomInt(s);
    }
}

```

```

        continue;
    }

    if (c == 'I') { // Xet them ki tu r[i] sat sau c: IV, IX
        s = Str(c);
        if (Cpos(r[i], "VX") >= 0) {
            s += r[i];
            ++i;
        }
        val += RomInt(s);
        continue;
    }
} // for
return val;
} // ToInt

main() {
    cout << "\n Result = " << ToInt("MCMXLVI");
    cout << "\n T H E      E N D";
    return 0;
}

```

## Roman Numerals Version 2: Roman To Integer and vise versa

```

/*****
    Roman Numerals (Ver. 2)
    ToInt:  Roman -> Integer
    ToRom: Inter ->  Roman
*****/
#include <iostream>
#include <cmath>
using namespace std;

int ival[] = {1000, 900, 500, 400, 100, 90, 50, 40, 10, 9, 5, 4, 1};
string rval[] = {"M", "CM", "D", "CD", "C", "XC", "L", "XL", "X", "IX", "V", "IV", "I"};
int len = 13;

void Go() {
    cout << " ? ";
    fflush(stdin);
    if (cin.get() == '.')
        exit(0);
}

// Vi tri xuat hien ki tu c trong string s
int Cpos(char c, string s) {
    for (int i = 0; i < s.length(); ++i)
        if (s[i] == c) return i;
    return -1;
} // Cpos

// Vi tri xuat hien string r trong rval
int Rpos(string r) {
    for (int i = 0; i < len; ++i)
        if (rval[i] == r) return i;
}

```

```

    return -1;
} // Rpos

// Vi tri xuat hien int v trong ival
int Ipos(int v) {
    for (int i = 0; i < len; ++i)
        if (ival[i] == v) return i;
    return -1;
} // Ipos

// rom r -> int
int RomInt(string r) {
    int i = Rpos(r);
    return (i < 0) ? 0 : ival[i];
} // RomInt

// int v -> rom
string IntRom(int v) {
    int i = Ipos(v);
    return (i < 0) ? "" : rval[i];
} // IntRom

string Str(char c) {
    string s = "";
    s += c;
    return s;
} // Str

int ToInt(string inpr) { // convert rom to int CDLVI -> 456
    // Bo cac dau cach
    string r = "";
    for (int i = 0; i < inpr.length(); ++i)
        if (inpr[i] != 32) r += inpr[i];

    if (r == "") return 0;

    r += "#"; // Them linh canh cuoi xau: guards
    int n = r.length();
    int val = 0; // output value
    int i = 0;
    string s;

    while (i < n) {
        char c = r[i]; // ki tu hien hanh
        i++; // ki tu sat sau c

        if (Cpos(c, "MDLV") >= 0) {
            val += RomInt(Str(c));
            continue;
        }

        if (c == 'C') { // Xet them ki tu r[i] sat sau c: CM, CD
            s = Str(c);
            if (Cpos(r[i], "MD") >= 0) {
                s += r[i];
                i++;
            }
        }
    }
}

```

```

        val += RomInt(s);
        continue;
    }

    if (c == 'X') { // Xet them ki tu r[i] sat sau c: XL, XC
        s = Str(c);
        if (Cpos(r[i], "LC") >= 0) {
            s += r[i];
            ++i;
        }
        val += RomInt(s);
        continue;
    }

    if (c == 'I') { // Xet them ki tu r[i] sat sau c: IV, IX
        s = Str(c);
        if (Cpos(r[i], "VX") >= 0) {
            s += r[i];
            ++i;
        }
        val += RomInt(s);
        continue;
    }
} // for
return val;
} // ToInt

string ToRom(int n) { // convert int to rom: 456 -> CDLVI
    string r = "";
    for (int i = 0; i < len; ++i) {
        while (n >= ival[i]) {
            r += IntRom(ival[i]);
            n -= ival[i];
        } // while
    } // for
    return r;
} // ToRom

main() {
    // cout << "\n Result = " << ToInt("MCMXLVI");
    int n = 3946;
    cout << "\n " << n << " -> " << ToRom(n);
    cout << "\n T H E    E N D";
    return 0;
}

```

### Roman Numerals (Ver. 3)

```

/*****
    Roman Numerals (Ver. 3)
    Improve

*****/
#include <iostream>
#include <cmath>

```

```

using namespace std;

int ival[] =
{1000, 900, 500, 400,100, 90, 50, 40, 10, 9, 5, 4, 1};

string rval[]=
{"M", "CM","D", "CD","C", "XC", "L", "XL", "X", "IX", "V", "IV", "I"};
int len = 13;

void Go() {
    cout << " ? ";
    fflush(stdin);
    if (cin.get() == '.')
        exit(0);
} // Go

string Str(char c) {
    string s = "";
    s += c;
    return s;
} // Str

// Vi tri xuat hien string r trong rval
int Rpos(string r) {
    for (int i = 0; i < len; ++i)
        if (rval[i] == r) return i;
    return -1;
} // Rpos

// Vi tri xuat hien int v trong ival
int Ipos(int v) {
    for (int i = 0; i < len; ++i)
        if (ival[i] == v) return i;
    return -1;
} // Ipos

// rom r -> int
int RomInt(string r) {
    int i = Rpos(r);
    return (i < 0) ? 0 : ival[i];
} // RomInt

// int v -> rom
string IntRom(int v) {
    int i = Ipos(v);
    return (i < 0) ? "" : rval[i];
} // IntRom

// CM = 900, CD = 40; XC = 90, XL = 40; IX = 9, IV = 4
int Pair(char c, char cc, int &d) {
    string s = Str(c);
    switch(c) {
        case 'C': // CM | CD
            if (cc == 'M' || cc == 'D') s += cc;
            break;
        case 'X': // XC | XL

```

```

        if (cc == 'C' || cc == 'L') s += cc;
        break;
    case 'I': // IX | IV
        if (cc == 'X' || cc == 'V') s += cc;
        break;
    } // switch
    d = s.length();
    return RomInt(s);
} // Pair

int ToInt(string inpr) { // convert rom to int CDLVI -> 456
    // Bo cac dau cach: inpr -> r
    string r = "";
    for (int i = 0; i < inpr.length(); ++i)
        if (inpr[i] != 32) r += inpr[i];
    // if (r == "") return 0;
    r += "#"; // Them linh canh cuoi xau: guards
    int val = 0; // output value
    int d = 0;
    for (int i = 0; i < r.length(); i += d) {
        val += Pair(r[i], r[i+1], d);
    } // for
    return val;
} // ToInt

string ToRom(int n) { // convert int to rom: 456 -> CDLVI
    string r = "";
    for (int i = 0; i < len; ++i) {
        while (n >= ival[i]) {
            r += IntRom(ival[i]);
            n -= ival[i];
        } // while
    } // for
    return r;
} // ToRom

main() {
    cout << "\n Result = " << ToInt("AB"); // 0
    cout << "\n Result = " << ToInt(""); // 0
    cout << "\n Result = " << ToInt("MCMXLVI"); // 1946
    cout << "\n Result = " << ToRom(1946); // MCMXLVI
    cout << "\n Result = " << ToRom(0); //
    cout << "\n T H E   E N D";
    return 0;
}

```



## Roman Fibonacci

Write a function RFib(n) giving the n-th Fibonacci number.

### Example

f(1) = 1, f(2) = 1, f(3) = 2, f(4) = 3, f(5) = 5, f(6) = 8, f(7) = 13, ...

RFib("I") = "I", RFib("II") = "I", RFib("III") = "II", RFib("IV") = "III",

RFib("V") = "V", RFib("VI") = "VIII", RFib("VII") = "XIII", ...

Reuse is a nice action in the life

### Roman Ficonacci Version 1

```
/*
*****
    RFib (Ver. 1)
*****
#include <iostream>
#include <cmath>
using namespace std;

int ival[] =
{1000, 900, 500, 400,100, 90, 50, 40, 10, 9, 5, 4, 1};

string rval[]=
{"M", "CM","D", "CD","C", "XC", "L", "XL", "X", "IX", "V", "IV", "I"};
int len = 13;

void Go() {
    cout << " ? ";
    fflush(stdin);
    if (cin.get() == '.')
        exit(0);
} // Go

string Str(char c) {
    string s = "";
    s += c;
    return s;
} // Str

// Vi tri xuat hien string r trong rval
int Rpos(string r) {
    for (int i = 0; i < len; ++i)
        if (rval[i] == r) return i;
    return -1;
} // Rpos

// Vi tri xuat hien int v trong ival
```

```

int Ipos(int v) {
    for (int i = 0; i < len; ++i)
        if (ival[i] == v) return i;
    return -1;
} // Ipos

// rom r -> int
int RomInt(string r) {
    int i = Rpos(r);
    return (i < 0) ? 0 : ival[i];
} // RomInt

// int v -> rom
string IntRom(int v) {
    int i = Ipos(v);
    return (i < 0) ? "" : rval[i];
} // IntRom

// CM = 900, CD = 40; XC = 90, XL = 40; IX = 9, IV = 4
int Pair(char c, char cc, int &d) {
    string s = Str(c);
    switch(c) {
        case 'C': // CM | CD
            if (cc == 'M' || cc == 'D') s += cc;
            break;
        case 'X': // XC | XL
            if (cc == 'C' || cc == 'L') s += cc;
            break;
        case 'I': // IX | IV
            if (cc == 'X' || cc == 'V') s += cc;
            break;
    } // switch
    d = s.length();
    return RomInt(s);
} // Pair

int ToInt(string inpr) { // convert rom to int CDLVI -> 456
    // Bo cac dau cach: inpr -> r
    string r = "";
    for (int i = 0; i < inpr.length(); ++i)
        if (inpr[i] != 32) r += inpr[i];
    // if (r == "") return 0;
    r += "#"; // Them linh canh cuoi xau: guards
    int val = 0; // output value
    int d = 0;
    for (int i = 0; i < r.length(); i += d) {
        val += Pair(r[i], r[i+1], d);
    } // for
    return val;
} // ToInt

string ToRom(int n) { // convert int to rom: 456 -> CDLVI
    string r = "";
    for (int i = 0; i < len; ++i) {
        while (n >= ival[i]) {
            r += IntRom(ival[i]);
            n -= ival[i];
        }
    }
}

```

```

        } // while
    } // for
    return r;
} // ToRom

int Fib(int n) {
    int a = 1;
    int b = 1;
    int c;
    for (int i = 3; i <= n; ++i) {
        c = a + b;
        a = b;
        b = c;
    } // for
    return b;
} // Fib

string RFib(string r) {
    return ToRom(Fib(ToInt(r)));
} // RFib

void Test() {
    int nn = 20;
    for (int n = 1; n <= nn; ++n)
        cout << "\n n = " << n << "   Fib = " << Fib(n) << "   RFib = " <<
RFib(ToRom(n));

} // Test

main() {
    Test();
    cout << "\n T H E   E N D";
    return 0;
}

```

## Roman Ficonacci Version 2 (Preparing)

Preparing is a important step

```

/*****
    Roman Fibonacci (Ver. 2)
    Preparing

*****/
#include <iostream>
#include <cmath>
using namespace std;

typedef long long Long;

int ival[] =

```

```

{1000, 900, 500, 400, 100, 90, 50, 40, 10, 9, 5, 4, 1};

string rval[] =
{"M", "CM", "D", "CD", "C", "XC", "L", "XL", "X", "IX", "V", "IV", "I"};
int len = 13;

const int MN = 90;
int f[MN+1];

void Go() {
    cout << " ? ";
    fflush(stdin);
    if (cin.get() == '.')
        exit(0);
} // Go

string Str(char c) {
    string s = "";
    s += c;
    return s;
} // Str

// Vi tri xuat hien string r trong rval
int Rpos(string r) {
    for (int i = 0; i < len; ++i)
        if (rval[i] == r) return i;
    return -1;
} // Rpos

// Vi tri xuat hien int v trong ival
int Ipos(int v) {
    for (int i = 0; i < len; ++i)
        if (ival[i] == v) return i;
    return -1;
} // Ipos

// rom r -> int
Long RomInt(string r) {
    int i = Rpos(r);
    return (i < 0) ? 0 : ival[i];
} // RomInt

// int v -> rom
string IntRom(Long v) {
    int i = Ipos(v);
    return (i < 0) ? "" : rval[i];
} // IntRom

// CM = 900, CD = 40; XC = 90, XL = 40; IX = 9, IV = 4
int Pair(char c, char cc, int &d) {
    string s = Str(c);
    switch(c) {
        case 'C': // CM | CD
            if (cc == 'M' || cc == 'D') s += cc;
            break;
        case 'X': // XC | XL

```

```

        if (cc == 'C' || cc == 'L') s += cc;
        break;
    case 'I': // IX | IV
        if (cc == 'X' || cc == 'V') s += cc;
        break;
    } // switch
    d = s.length();
    return RomInt(s);
} // Pair

Long ToInt(string inpr) { // convert rom to int CDLVI -> 456
    // Bo cac dau cach: inpr -> r
    string r = "";
    for (int i = 0; i < inpr.length(); ++i)
        if (inpr[i] != 32) r += inpr[i];
    // if (r == "") return 0;
    r += "#"; // Them linh canh cuoi xau: guards
    Long val = 0; // output value
    int d = 0;
    for (int i = 0; i < r.length(); i += d) {
        val += Pair(r[i], r[i+1], d);
    } // for
    return val;
} // ToInt

string ToRom(Long n) { // convert int to rom: 456 -> CDLVI
    string r = "";
    for (int i = 0; i < len; ++i) {
        while (n >= ival[i]) {
            r += IntRom(ival[i]);
            n -= ival[i];
        } // while
    } // for
    return r;
} // ToRom

/*-----
                        Fibonacci
-----*/

int AllFib() {
    Long a, b, c;
    a = b = 1;
    f[1] = f[2] = 1;
    for (int i = 3; i <= MN; ++i) {
        c = a + b;
        f[i] = c;
        a = b;
        b = c;
    } // for
} // AllFib

Long Fib(int n) {
    return (n <= MN) ? f[n] : 0;
} // Fib

string RFib(string r) {

```

```

    int i = ToInt(r);
    return (i <= MN) ? ToRom(f[i]) : "";
} // RFib

void Test() {
    int nn = 20;
    for (int n = 1; n <= nn; ++n)
        cout << "\n n = " << n << "   Fib = "
            << Fib(n) << "   RFib = " << RFib(ToRom(n));
} // Test

void Test1() {
    int nn = 25; // 90
    AllFib();
    for (int n = 1; n <= nn; ++n) {
        cout << "\n n = " << n << "   Fib = "
            << Fib(n) << "   RFib = " << RFib(ToRom(n));
        Go();
    } // for
} // Test1

main() {
    // Test();
    Test1();
    cout << "\n T H E   E N D";
    return 0;
}

```

## Expression

Tính trị của biểu thức chứa các biến a..z, các phép toán +, -, \*, / với quy ước các biến được gán trị ngầm định a = 0, b = 1, ..., z = 25.

Ví dụ  $(b+d)*(f-c) = (1+3)*(4-2) = 4*2 = 8$ .

Programs = Algorithms + Data Structures  
Bàn tiệc = Cách làm + Tổ chức nguyên liệu

### Expression Version 1. (Simplest case)

```
/******  
    Expression (Ver. 1, Aug 8 2020)  
    + - * /  
******/  
#include <iostream>  
using namespace std;  
  
string OP = "+-*/";  
  
void Go() {  
    cout << " ? ";  
    fflush(stdin);  
    if (cin.get() == '.')  
        exit(0);  
}  
  
bool Var(char c) {  
    return 'a' <= c && c <= 'z';  
} // Var  
  
bool Op(char c) {  
    for (int i = 0; i < OP.length(); ++i)  
        if (OP[i] == c) return true;  
    return false;  
} // Op  
  
int Degree(char c) {  
    switch(c) {  
        case '+': return 500;  
        case '-': return 500;  
        case '*': return 100;  
        case '/': return 100;  
        default: return 1000;  
    } // switch  
} // Degree  
  
string Compiler(string inp) {
```

```

string out = "";
int n = inp.length();
char st[n];
char c;
int p = 0;
st[p] = '#'; // phan tu dem
for (int i = 0; i < n; ++i) {
    c = inp[i];
    if (Var(c)) { // Gap bien: dua vao out
        out += c;
        continue;
    }
    if (c == '(') { // Gap (: nap st
        st[++p] = c;
        continue;
    }
    if (Op(c)) { // Gap phep toan
        // Ro cac phep toan uu tien dua vao out
        while(Degree(st[p]) <= Degree(c))
            out += st[p--];
        st[++p] = c; // Nap phep toan moi vao st
        continue;
    }
    if (c == ')') { // Gap ):
        // Ro cac phep toan truoc ( dua vao out
        while(st[p] != '(')
            out += st[p--];
        st[p--]; // Ro (
    }
} // for
// Ro cac phep toan con lai dua vao out
while(st[p] != '#')
    out += st[p--];
cout << inp << " -> " << out; // Polish form (Lukasievics)
return out;
}

int Exe(string e){
    int st[e.length()];
    int p = -1;
    char c;
    // Nap x vao st (Push): st[++p] = x
    // Lay ngon st ra khoi st dua vao x: (Pop): x = st[p--]
    for (int i = 0; i <= e.length(); ++i) {
        c = e[i];
        if (Var(c)) {
            st[++p] = c-'a'; // nap st
            continue;
        }
        if (Op(c)) {
            // thuc hien phep toan c
            switch(c) {
                case '+': st[p-1] += st[p];
                        --p;
                        break;
                case '-': st[p-1] -= st[p];
                        --p;
            }
        }
    }
}

```



```

        break;
    case '*': st[p-1] *= st[p];
        --p;
        break;
    case '/': st[p-1] /= st[p];
        --p;
        break;

    } // switch
    continue;
} // Op
} // for
return st[0] ;
} // Exe

main() {
    // string pol = Compiler("(a+c)*(g - b)"); // (0+2)*(6-1) = 10
    string pol = Compiler("(a+c)*(g - b) / (g - b) "); // 2
    cout << "\n Result: " << Exe(pol);
    cout << "\n T H E      E N D";
    return 0;
}

```

## Expression Version 2. (Error messages)

```

/*****
Expression (Ver. 2, Aug 8 2020)
+ - * /
Error messages

*****/
#include <iostream>
using namespace std;

string OP = "+-*/";

void Go() {
    cout << " ? ";
    fflush(stdin);
    if (cin.get() == '.')
        exit(0);
}

void Err(int e) {
    cerr << "\n Error " << e << ": ";
    switch(e) {
        case 1: cout << "( expected.";
            exit(0);
        case 2: cout << "Syntax error.";
            exit(0);
        case 3: cout << " devide by zero.";
            exit(0);
    } // switch
} // Err

```

```

bool Var(char c) {
    return 'a' <= c && c <= 'z';
} // Var

bool Op(char c) {
    for (int i = 0; i < OP.length(); ++i)
        if (OP[i] == c) return true;
    return false;
} // Op

int Degree(char c) {
    switch(c) {
        case '+': return 500;
        case '-': return 500;
        case '*': return 100;
        case '/': return 100;
        default: return 1000;
    } // switch
} // Degree

string Compiler(string inp) {
    string out = "";
    int n = inp.length();
    char st[n];
    char c;
    int p = 0;
    st[p] = '#'; // phan tu dem
    for (int i = 0; i < n; ++i) {
        c = inp[i];
        if (Var(c)) { // Gap bien: dua vao out
            out += c;
            continue;
        }
        if (c == '(') { // Gap (: nap st
            st[++p] = c;
            continue;
        }
        if (Op(c)) { // Gap phep toan
            // Ro cac phep toan uu tien dua vao out
            while(Degree(st[p]) <= Degree(c))
                out += st[p--];
            st[++p] = c; // Nap phep toan moi vao st
            continue;
        }
        if (c == ')') { // Gap ):
            // Ro cac phep toan truoc ( dua vao out
            while(st[p] != '(') {
                if (st[p] == '#') Err(1);
                out += st[p--];
            }
            st[p--]; // Ro (
            continue;
        }
        if (c == 32) continue;
        cout << " ??? " << c; Err(2);
    } // for
}

```

```

// Ro cac phep toan con lai dua vao out
while(st[p] != '#')
    out += st[p--];
cout << inp << " -> " << out; // Polish form (Lukasievics)
return out;
}

int Exe(string e) {
    int st[e.length()];
    int p = -1;
    char c;
    // Nap x vao st (Push): st[++p] = x
    // Lay ngon st ra khoi st dua vao x: (Pop): x = st[p--]
    for (int i = 0; i <= e.length(); ++i) {
        c = e[i];
        if (Var(c)) {
            st[++p] = c-'a'; // nap st
            continue;
        }
        if (Op(c)) {
            // thuc hien phep toan c
            switch(c) {
                case '+':
                    if (p < 1) Err(2);
                    st[p-1] += st[p];
                    --p;
                    break;
                case '-': if (p < 1) Err(2);
                    st[p-1] -= st[p];
                    --p;
                    break;
                case '*': if (p < 1) Err(2);
                    st[p-1] *= st[p];
                    --p;
                    break;
                case '/': if (p < 1) Err(2);
                    if (st[p] == 0) Err(3);
                    st[p-1] /= st[p];
                    --p;
                    break;
            } // switch
            continue;
        } // Op
    } // for
    return st[0];
} // Exe

main() {
    // string pol = Compiler("(a+c)*(g - b)"); // (0+2)*(6-1) = 10
    string pol = Compiler("(a+c)*+(g - b) / (g - b) "); // 2
    cout << "\n Result: " << Exe(pol);
    cout << "\n T H E    E N D";
    return 0;
}

```

### Expression Version 3. (More operators)

```

/*****
Expression (Ver. 3, Aug 8 2020)
+ - * /
Them phep toan:
%(chia du), >(tang 1), <(giam 1), !(giai thua)
^(luy thua)

*****/
#include <iostream>
#include <cmath>
using namespace std;

string OP = "+-*/%><!^";

void Go() {
    cout << " ? ";
    fflush(stdin);
    if (cin.get() == '.')
        exit(0);
}

void Err(int e) {
    cerr << "\n Error " << e << ": ";
    switch(e) {
        case 1: cout << "( expected.";
                exit(0);
        case 2: cout << "Syntax error.";
                exit(0);
        case 3: cout << " devide by zero.";
                exit(0);
        case 4: cout << " negative factorial.";
                exit(0);
    } // switch
} // Err

bool Var(char c) {
    return 'a' <= c && c <= 'z';
} // Var

bool Op(char c) {
    for (int i = 0; i < OP.length(); ++i)
        if (OP[i] == c) return true;
    return false;
} // Op

int Degree(char c) {
    switch(c) {
        case '^':
        case '!':
        case '>':
    }
}

```

```

        case '<': return 100;
        case '*':
            case '%':
        case '/': return 200;
        case '+':
        case '-': return 500;
        default: return 1000;
    } // switch
} // Degree

string Compiler(string inp) {
    string out = "";
    int n = inp.length();
    char st[n];
    char c;
    int p = 0;
    st[p] = '#'; // phan tu dem
    for (int i = 0; i < n; ++i) {
        c = inp[i];
        if (Var(c)) { // Gap bien: dua vao out
            out += c;
            continue;
        }
        if (c == '(') { // Gap (: nap st
            st[++p] = c;
            continue;
        }
        if (Op(c)) { // Gap phep toan
            // Ro cac phep toan uu tien dua vao out
            while(Degree(st[p]) <= Degree(c))
                out += st[p--];
            st[++p] = c; // Nap phep toan moi vao st
            continue;
        }
        if (c == ')') { // Gap ):
            // Ro cac phep toan truoc ( dua vao out
            while(st[p] != '(') {
                if (st[p] == '#') Err(1);
                out += st[p--];
            }
            st[p--]; // Ro (
            continue;
        }
        if (c == 32) continue;
        cout << " ??? " << c; Err(2);
    } // for
    // Ro cac phep toan con lai dua vao out
    while(st[p] != '#')
        out += st[p--];
    cout << inp << " -> " << out; // Polish form (Lukasievics)
    return out;
}

int Fac(int n) {
    int r = 1;
    if (n < 0) Err(4);
    for (int i = 2; i <= n; ++i)

```

```

        r *= i;
    return r;
} // fac

int Exe(string e) {
    int st[e.length()];
    int p = -1;
    char c;
    // Nap x vao st (Push): st[++p] = x
    // Lay ngon st ra khoi st dua vao x: (Pop): x = st[p--]
    for (int i = 0; i <= e.length(); ++i) {
        c = e[i];
        if (Var(c)) {
            st[++p] = c-'a'; // nap st
            continue;
        }
        if (Op(c)) {
            // thuc hien phep toan c
            switch(c) {
                case '>': if (p < 0) Err(2);
                        ++st[p] ;
                        break;
                case '<': if (p < 0) Err(2);
                        --st[p] ;
                        break;
                case '!': if (p < 0) Err(2);
                        st[p] = Fac(st[p]);
                        break;
                case '^': if (p < 1) Err(2);
                        st[p-1] = pow(st[p-1],st[p]);
                        --p;
                        break;
                case '*': if (p < 1) Err(2);
                        st[p-1] *= st[p];
                        --p;
                        break;

                case '/': if (p < 1) Err(2);
                        if (st[p] == 0) Err(3);
                        st[p-1] /= st[p];
                        --p;
                        break;

                case '%': if (p < 1) Err(2);
                        if (st[p] == 0) Err(3);
                        st[p-1] %= st[p];
                        --p;
                        break;
                case '+':
                        if (p < 1) Err(2);
                        st[p-1] += st[p];
                        --p;
                        break;
                case '-': if (p < 1) Err(2);
                        st[p-1] -= st[p];
                        --p;
                        break;
            }
        }
    }
}

```

```

        } // switch
        continue;
    } // Op
} // for
return st[0];
} // Exe

main() {
    // abcdefghijklmnopqrstuvwxyz
    // 01234567891012345678912345
    // string pol = Compiler("(a+c)*(g - b)"); // (0+2)*(6-1) = 10
    // string pol = Compiler("(a+c)+(g - b) / (g - b) "); // 2
    // string pol = Compiler("(a>+c>!)*(g - b) / (g - b))");
    // (0>+2>!)*(6-1)/(6-1) = (1+6)*5/5 = 7
    // string pol = Compiler("(a>+c>!)*(g - b) / (g - b))! ");
    // 7! = 1.2.3.4.5.6.7 = 5040
    string pol = Compiler("k*c^k");
    // 10*2^10 = 10*(2^10) = 10*1024 = 10240
    cout << "\n Result: " << Exe(pol);
    cout << "\n T H E    E N D";
    return 0;
}

```

#### Expression Version 4. (Negative operator)

```

/*****
Expression (Ver. 4, Aug 8 2020)
Doi dau:
-(c+e) -> ce+~
c*(-e+f) = ce~f+*

*****/
#include <iostream>
#include <cmath>
using namespace std;

string OP = "+-*/%><!^~";

void Go() {
    cout << " ? ";
    fflush(stdin);
    if (cin.get() == '.')
        exit(0);
}

void Err(int e) {
    cerr << "\n Error " << e << ": ";
    switch(e) {
        case 1: cout << "( expected.";
                exit(0);
        case 2: cout << "Syntax error.";
                exit(0);
        case 3: cout << " devide by zero.";
    }
}

```

```

        exit(0);
    case 4: cout << " negative factorial.";
        exit(0);

    } // switch
} // Err

bool Var(char c) {
    return 'a' <= c && c <= 'z';
} // Var

bool Op(char c) {
    for (int i = 0; i < OP.length(); ++i)
        if (OP[i] == c) return true;
    return false;
} // Op

int Degree(char c) {
    switch(c) {
        case '~':
        case '^':
        case '!':
        case '>':
        case '<': return 100;
        case '*':
        case '%':
        case '/': return 200;
        case '+':
        case '-': return 500;
        default: return 1000;
    } // switch
} // Degree

string Compiler(string inps) {
    string inp = "";
    for (int i = 0; i < inps.length(); ++i)
        if (inps[i] != 32) inp += inps[i];
    string out = "";
    int n = inp.length();
    char st[n];
    char c, cc;
    int p = 0;
    st[p] = '#'; // phan tu dem
    for (int i = 0; i < n; ++i) {
        c = inp[i];
        if (Var(c)) { // Gap bien: dua vao out
            out += c;
            continue;
        }
        if (c == '(') { // Gap (: nap st
            st[++p] = c;
            continue;
        }
        if (c == '+' || c == '-') {
            if(i == 0)
                cc = (c == '-') ? '~' : '.';
            else if (inp[i - 1] == '(')

```



```

        cc = (c == '-') ? '~': '.';
        else cc = c;
        if (cc == '.') continue;
        while (Degree(st[p]) <= Degree(cc))
            out += st[p--];
        st[++p] = cc;
        continue;
    }
    if (Op(c)) { // Gap phep toan
        // Ro cac phep toan uu tien dua vao out
        while(Degree(st[p]) <= Degree(c))
            out += st[p--];
        st[++p] = c; // Nap phep toan moi vao st
        continue;
    }
    if (c == ')') { // Gap ):
        // Ro cac phep toan truoc ( dua vao out
        while(st[p] != '(') {
            if (st[p] == '#') Err(1);
            out += st[p--];
        }
        st[p--]; // Ro (
        continue;
    }
    if (c == 32) continue;
    cout << " ??? " << c; Err(2);
} // for
// Ro cac phep toan con lai dua vao out
while(st[p] != '#')
    out += st[p--];
cout << inp << " -> " << out; // Polish form (Lukasievics)
return out;
}

int Fac(int n) {
    int r = 1;
    if (n < 0) Err(4);
    for (int i = 2; i <= n; ++i)
        r *= i;
    return r;
} // fac

int Exe(string e) {
    int st[e.length()];
    int p = -1;
    char c;
    // Nap x vao st (Push): st[++p] = x
    // Lay ngon st ra khoi st dua vao x: (Pop): x = st[p--]
    for (int i = 0; i <= e.length(); ++i) {
        c = e[i];
        if (Var(c)) {
            st[++p] = c-'a'; // nap st
            continue;
        }
        if (Op(c)) {
            // thuc hien phep toan c
            switch(c) {

```

```

        case '~': if (p < 0) Err(2);
                  st[p] = -st[p];
                  break;
        case '>': if (p < 0) Err(2);
                  ++st[p] ;
                  break;
        case '<': if (p < 0) Err(2);
                  --st[p] ;
                  break;
        case '!': if (p < 0) Err(2);
                  st[p] = Fac(st[p]);
                  break;
        case '^': if (p < 1) Err(2);
                  st[p-1] = pow(st[p-1],st[p]);
                  --p;
                  break;
        case '*': if (p < 1) Err(2);
                  st[p-1] *= st[p];
                  --p;
                  break;

        case '/': if (p < 1) Err(2);
                  if (st[p] == 0) Err(3);
                  st[p-1] /= st[p];
                  --p;
                  break;

        case '%': if (p < 1) Err(2);
                  if (st[p] == 0) Err(3);
                  st[p-1] %= st[p];
                  --p;
                  break;
        case '+':
                  if (p < 1) Err(2);
                  st[p-1] += st[p];
                  --p;
                  break;
        case '-': if (p < 1) Err(2);
                  st[p-1] -= st[p];
                  --p;
                  break;

    } // switch
    continue;
} // Op
} // for
return st[0];
} // Exe

main() {
    // a b c d e f g h i j k l m n o p q r s t u v w x y z
    // 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
    // string pol = Compiler("(a+c)*(g - b)"); // (0+2)*(6-1) = 10
    // string pol = Compiler("(a+c)+(g - b) / (g - b) "); // 2
    // string pol = Compiler("((a>!+c>!)*(g - b) / (g - b))");
    // (0>!+2>!)*(6-1)/(6-1) = (1+6)*5/5 = 7

```

```

// string pol = Compiler("((a>!+c>!)*(g - b) / (g - b))! ");
// 7! = 1.2.3.4.5.6.7 = 5040
// string pol = Compiler("k*c^k");
// 10*2^10 = 10*(2^10) = 10*1024 = 10240
string pol = Compiler("-(+c * e * (-c + g)) ");
// -(+2 * 4 * (-2 + 6)) = -32
cout << "\n Result: " << Exe(pol);
cout << "\n T H E   E N D";
return 0;
}

```

## Expression Version 5. (Roman Numerals)

```

/*****
Expression (Ver. 5, Aug 8 2020)
Roman and Integers

*****/
#include <iostream>
#include <cmath>
using namespace std;

void Go() {
    cout << " ? ";
    fflush(stdin);
    if (cin.get() == '.')
        exit(0);
}

/*-----
                        Roman numerals
-----*/
int ival[] = {1000, 900, 500, 400, 100, 90, 50, 40, 10, 9, 5, 4, 1};
string rval[] = {"M", "CM", "D", "CD", "C", "XC", "L", "XL", "X", "IX", "V", "IV", "I"};
int len = 13;
string ROM = "MDCLXVI";

// Vi tri xuat hien ki tu c trong string s
int Cpos(char c, string s) {
    for (int i = 0; i < s.length(); ++i)
        if (s[i] == c) return i;
    return -1;
} // Cpos

// Vi tri xuat hien string r trong rval
int Rpos(string r) {
    for (int i = 0; i < len; ++i)
        if (rval[i] == r) return i;
    return -1;
} // Rpos

// Vi tri xuat hien int v trong ival
int Ipos(int v) {

```

```

    for (int i = 0; i < len; ++i)
        if (ival[i] == v) return i;
    return -1;
} // Ipos

// rom r -> int
int RomInt(string r) {
    int i = Rpos(r);
    return (i < 0) ? 0 : ival[i];
} // RomInt

// int v -> rom
string IntRom(int v) {
    int i = Ipos(v);
    return (i < 0) ? "" : rval[i];
} // IntRom

string Str(char c) {
    string s = "";
    s += c;
    return s;
} // Str

int ToInt(string inpr) { // convert rom to int CDLVI -> 456
    // Bo cac dau cach
    string r = "";
    for (int i = 0; i < inpr.length(); ++i)
        if (inpr[i] != 32) r += inpr[i];

    if (r == "") return 0;

    r += "#"; // Them linh canh cuoi xau: guards
    int n = r.length();
    int val = 0; // output value
    int i = 0;
    string s;

    while (i < n) {
        char c = r[i]; // ki tu hien hanh
        i++; // ki tu sat sau c

        if (Cpos(c, "MDLV") >= 0) {
            val += RomInt(Str(c));
            continue;
        }

        if (c == 'C') { // Xet them ki tu r[i] sat sau c: CM, CD
            s = Str(c);
            if (Cpos(r[i], "MD") >= 0) {
                s += r[i];
                i++;
            }
            val += RomInt(s);
            continue;
        }

        if (c == 'X') { // Xet them ki tu r[i] sat sau c: XL, XC

```

```

        s = Str(c);
        if (Cpos(r[i],"LC") >= 0) {
            s += r[i];
            ++i;
        }
        val += RomInt(s);
        continue;
    }

    if (c == 'I') { // Xet them ki tu r[i] sat sau c: IV, IX
        s = Str(c);
        if (Cpos(r[i],"VX") >= 0) {
            s += r[i];
            ++i;
        }
        val += RomInt(s);
        continue;
    }
} // for
return val;
} // ToInt

string ToRom(int n) { // convert int to rom: 456 -> CDLVI
    string r = "";
    for (int i = 0; i < len; ++i) {
        while (n >= ival[i]) {
            r += IntRom(ival[i]);
            n -= ival[i];
        } // while
    } // for
    return r;
} // ToRom

/*-----
                        Expression
-----*/

string OP = "+-*/%><!^~";
int cval[1000]; // constants
int clen = 0;

void Err(int e) {
    cerr << "\n Error " << e << ": ";
    switch(e) {
        case 1: cout << "( expected.";
                exit(0);
        case 2: cout << "Syntax error.";
                exit(0);
        case 3: cout << " devide by zero.";
                exit(0);
        case 4: cout << " negative factorial.";
                exit(0);

    } // switch
} // Err

bool Var(char c) {
    return 'a' <= c && c <= 'z';
}

```

```

} // Var

bool Num(char c) {
    return '0' <= c && c <= '9';
} // Num

bool Rom(char c) {
    return (Cpos(c,ROM) >= 0);
} // Rom

bool Op(char c) {
    for (int i = 0; i < OP.length(); ++i)
        if (OP[i] == c) return true;
    return false;
} // Op

int Degree(char c) {
    switch(c) {
        case '~':
        case '^':
        case '!':
        case '>':
        case '<': return 100;
        case '*':
        case '%':
        case '/': return 200;
        case '+':
        case '-': return 500;
        default: return 1000;
    } // switch
} // Degree

string Precomp(string s) {

    string out = "";
    // string x;
    int d;
    int v;
    clen = 0;
    for (int i = 0; i < s.length(); i += d) {
        d = 1;

        if (s[i] == 32) continue;

        if (Var(s[i]) || Op(s[i]) || s[i] == '(' || s[i] == ')') {
            out += s[i];
            continue;
        }

        if (Num(s[i])) {
            v = 0;
            for (int j = i; j < s.length(); ++j) {
                if (Num(s[j]))
                    v = v * 10 + (s[j] - '0');
                else {
                    d = j-i;
                    break;
                }
            }
        }
    }
}

```

```

        }
    } // for j
    out += "$";
    cval[clen++] = v;
    continue;
} // if Num

if (Rom(s[i])) {
    string x = "";
    for (int j = i; j < s.length(); ++j) {
        if (Rom(s[j]))
            x += s[j];
        else {
            d = j-i;
            break;
        }
    }
    out += "$";
    cval[clen++] = ToInt(x);

    continue;
} // if Rom
Err(2);
} // for
cout << "\n Precomp: " << s << " -> " << out;
cout << "\n Const = ";
for (int i = 0; i < clen; ++i)
    cout << " " << cval[i];
return out;
} // Precomp

string Compiler(string inps) {
    string inp = Precomp(inps);
    string out = "";
    int n = inp.length();
    char st[n];
    char c, cc;
    int p = 0;
    st[p] = '#'; // phan tu dem
    // cout << "\n Compiler " << inp;

    for (int i = 0; i < n; ++i) {
        c = inp[i];
        if (Var(c) || c == '$') { // Gap bien hoac '$': dua vao out
            out += c;
            continue;
        }
        if (c == '(') { // Gap (: nap st
            st[++p] = c;
            continue;
        }
        if (c == '+' || c == '-') {
            if (i == 0)
                cc = (c == '-') ? '~' : '.';
            else if (inp[i - 1] == '(')
                cc = (c == '-') ? '~' : '.';
            else cc = c;
        }
    }
}

```

```

        if (cc == '.') continue;
        while (Degree(st[p]) <= Degree(cc))
            out += st[p--];
        st[++p] = cc;
        continue;
    }
    if (Op(c)) { // Gap phép toán
        // Ro cac phép toán ưu tiên đưa vào out
        while(Degree(st[p]) <= Degree(c))
            out += st[p--];
        st[++p] = c; // Nạp phép toán mới vào st
        continue;
    }
    if (c == ')') { // Gap ):
        // Ro cac phép toán trước ( đưa vào out
        while(st[p] != '(') {
            if (st[p] == '#') Err(1);
            out += st[p--];
        }
        st[p--]; // Ro (
        continue;
    }
    cout << " ??? " << c; Err(2);
} // for
// Ro cac phép toán còn lại đưa vào out
while(st[p] != '#')
    out += st[p--];
cout << "\n Compiler: " << inp << " -> " << out; // Polish form
(Lukasievics)
return out;
}

int Fac(int n) {
    int r = 1;
    if (n < 0) Err(4);
    for (int i = 2; i <= n; ++i)
        r *= i;
    return r;
} // fac

int Exe(string e) {
    int st[e.length()];
    int p = -1;
    char c;
    int ic = -1; // index of cval
    // Nạp x vào st (Push): st[++p] = x
    // Lấy ngôn st ra khỏi st đưa vào x: (Pop): x = st[p--]
    for (int i = 0; i <= e.length(); ++i) {
        c = e[i];
        if (Var(c)) {
            st[++p] = c-'a'; // nạp st
            continue;
        }
        if (c == '$') { // hàng
            st[++p] = cval[++ic]; // nạp st
            continue;
        }
    }
}

```



```

if (Op(c)) {
    // thuc hien phep toan c
    switch(c) {

        case '~': if (p < 0) Err(2);
                  st[p] = -st[p];
                  break;
        case '>': if (p < 0) Err(2);
                  ++st[p] ;
                  break;
        case '<': if (p < 0) Err(2);
                  --st[p] ;
                  break;
        case '!': if (p < 0) Err(2);
                  st[p] = Fac(st[p]);
                  break;
        case '^': if (p < 1) Err(2);
                  st[p-1] = pow(st[p-1],st[p]);
                  --p;
                  break;
        case '*': if (p < 1) Err(2);
                  st[p-1] *= st[p];
                  --p;
                  break;

        case '/': if (p < 1) Err(2);
                  if (st[p] == 0) Err(3);
                  st[p-1] /= st[p];
                  --p;
                  break;

        case '%': if (p < 1) Err(2);
                  if (st[p] == 0) Err(3);
                  st[p-1] %= st[p];
                  --p;
                  break;
        case '+':
                  if (p < 1) Err(2);
                  st[p-1] += st[p];
                  --p;
                  break;
        case '-': if (p < 1) Err(2);
                  st[p-1] -= st[p];
                  --p;
                  break;

    } // switch
    continue;
} // Op
} // for
return st[0];
} // Exe

main() {
    // a b c d e f g h i j k l m n o p q r s t u v w x y z
    // 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

```

```

// string pol = Compiler("(a+c)*(g - b)"); // (0+2)*(6-1) = 10
// string pol = Compiler("(a+c)+(g - b) / (g - b) "); // 2
// string pol = Compiler("((a>!+c>!)*(g - b) / (g - b))");
// (0>!+2>!)*(6-1)/(6-1) = (1+6)*5/5 = 7
// string pol = Compiler("((a>!+c>!)*(g - b) / (g - b))! ");
// 7! = 1.2.3.4.5.6.7 = 5040
// string pol = Compiler("k*c^k"); // 10*2^10 = 10*(2^10) = 10*1024 =
10240
// string pol = Compiler("-(+c * e * (-c + g)) "); // -(+2 * 4 * (-2 +
6)) = -32
// string pol = Compiler(" (VI + e) *(-12 + c) ");
string pol = Compiler("-(+c * IV * (-II + g)) "); // -(+2 * 4 * (-2 +
6)) = -32
cout << "\n Result: " << Exe(pol); // 100
cout << "\n T H E    E N D";
return 0;
}

```



Xuất phát từ gốc.

Mỗi lần xét một ký hiệu  $b$  trong văn bản mật:

Nếu  $b = 0$  rẽ trái

Nếu  $b = 1$  rẽ phải

Nếu gặp lá: xuất ra ký tự trên lá rồi quay lại gốc.

### Biểu diễn cây

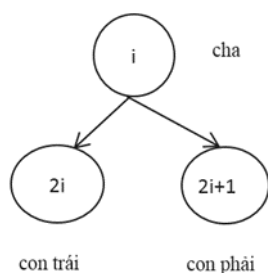
Chỉ cần một mảng tham chiếu. Ta gọi mảng đó là  $a$ . Ta gán trị  $a[v] = c$

trong đó  $v$  là số hiệu của (đỉnh) lá,  $c$  là ký tự trên lá. Ví dụ trên cho ta

v : Số hiệu đỉnh	...	8	9	...	11	12	...	14	15	...	20	21	...	26	27	...
Ký tự	#	A	C	#	M	O	#	V	*	#	E	H	#	T	U	#

trong đó dấu # biểu diễn cho giá trị trống.

Việc còn lại chỉ là xác định các số hiệu của các đỉnh.



Mỗi đỉnh  $i$  trong cây nhị phân có tối đa hai đỉnh con là con trái và con phải.  $i$  được gọi là đỉnh cha. Số hiệu của mỗi đỉnh được tính như sau:

Đỉnh cha:  $i$

Đỉnh con trái:  $2i$

Đỉnh con phải:  $2i+1$

Đỉnh gốc: 1

Điều thú vị là ta có thể tính trực tiếp số hiệu đỉnh treo cho mỗi ký tự theo mã của chúng theo cùng một thuật toán giống như thuật toán giải mã. Thật vậy, giả sử ký tự tại đỉnh treo trong cây có mã là  $M = m_1 m_2 \dots m_k$ .

Thuật toán xác định số hiệu  $v$   
của đỉnh treo

$v \leftarrow 1$

Mỗi lần xét một ký hiệu  $m_i$  trong mã ký tự:

Nếu  $m_i = 0$  :  $v \leftarrow 2 \cdot v$ ; // con trái

Nếu  $m_i = 1$  :  $v \leftarrow 2v + 1$ ; // con phải

Ví dụ

Giả sử ta cần xác định số hiệu  $v$  cho đỉnh treo (lá) ký tự T có mã 1010.

Ta đặt giá trị (1) cho  $v$  vào bên trái dãy mã 1010: (1)1010.

Tiếp theo ta dịch qua phải từng bit  $b$  của T và cập nhật  $v$  theo quy tắc :

( $v$ )0( $2v$ ): nếu  $b = 0$  thì cập nhật  $v \leftarrow 2v$

( $v$ )1( $2v+1$ ): nếu  $b = 1$  thì cập nhật  $v \leftarrow 2v+1$

.....

Cấu trúc dữ liệu và giải thuật C++ (Tom 1) tr. 44

Sơ đồ chi tiết của các bước được thể hiện như sau:

(1)1010 → 1(3)010 → 10(6)10 → 101(13)0 → 1010(26).

	T		T		T		T	
	b	v	b	v	b	v	b	v
(1)	1	(3)	0	(6)	1	(13)	0	(26)

$(v)0 \rightarrow 2v; (v)1 \rightarrow 2v+1$

TOMJERRY.INP	Giải thích
10	10 ký tự
A 000	mã số của A
C 001	mã số của C
E 0100	mã số của E
H 0101	mã số của H
M 011	mã số của M
O 100	mã số của O
T 1010	mã số của T
U 1011	mã số của U
V 110	mã số của V
* 111	mã số của *
0110100100111110000111001010110111001010	mã số của văn bản cần giải mã

Các bước của thuật toán tạo cây và giải mã sẽ như sau:

**Bước 1** Tạo cây

Khởi trị mảng a toàn ký tự '#'

Xác định số hiệu v cho mỗi ký tự c theo mã của chúng.

Gán trị  $a[v] \leftarrow c$

**Bước 2** Giải mã

Xuất phát từ gốc  $v \leftarrow 1$ .

Mỗi lần xét một ký hiệu b trong văn bản mật:

Nếu  $b = 0$ :  $v \rightarrow 2v$  (rẽ trái)

Nếu  $b = 1$ :  $v \rightarrow 2v+1$  (rẽ phải)

Nếu  $a[v] \neq \#$ : xuất  $a[v]$ ;  $v \rightarrow 1$  (xuất ký tự trên lá rồi quay lại gốc).

### Chương trình C++

```
// TomJerry.CPP
#include <iostream>
#include <fstream>
#include <windows.h>

using namespace std;

const int MN = 1024;
const char * fn = "TOMJERRY.INP";
```

```

int N; // so ky tu
char a[MN];

void run() {
    memset(a, '#', sizeof(a)); // Gan a[] toan #
    ifstream f(fn);
    string s;
    char c;
    int v;
    f >> N; // so ky tu
    for (int i = 1; i <= N; ++i) {
        f >> c >> s; // ky tu c co ma s
        // Xac dinh so hieu v cho ky tu c
        cout << "\n " << c << " : " << s;
        // lap ma so cho ky tu c
        v = 1;
        for (int j = 0; j < s.length(); ++j)
            v = (v * 2) + (s[j] - '0');
        a[v] = c;
        cout << " -> " << v;
    }
    // Doc doan code can giai ma
    f >> s;
    f.close();
    cout << "\n Decode " << s << "... \n ";
    v = 1;
    for (int i = 0; i < s.length(); ++i) {
        v = (v * 2) + (s[i] - '0');
        if (a[v] != '#') {
            cout << a[v];
            v = 1;
        }
    }
} // run

main() {
    run();
    cout << "\n T h e    E n d.";
    return 0;
}

```

#### Input file

```

TOMJERRY.INP
10
A 000
C 001
E 0100
H 0101
M 011
O 100
T 1010
U 1011
V 110
* 111
0110100100111110000111001010110111001010

```

## Kết quả

```
A : 000 -> 8
C : 001 -> 9
E : 0100 -> 20
H : 0101 -> 21
M : 011 -> 11
O : 100 -> 12
T : 1010 -> 26
U : 1011 -> 27
V : 110 -> 14
* : 111 -> 15
Decode 01101001001111110000111001010110111001010...
MEO*VA*CHUOT
T h e   E n d.
```

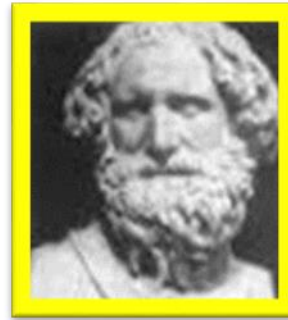
## Sieve of Eratosthenes

Vì phép nhân được thực hiện đơn giản hơn phép chia nên Eratosthenes, nhà toán học vĩ đại người Hy Lạp đã đề xuất ý tưởng tổ chức thuật toán tìm toàn bộ các số nguyên tố trong khoảng từ 1 đến giới hạn  $n$  cho trước. Người đời sau gọi thuật toán này là Sàng Eratosthenes. Bảng dưới minh họa hoạt động của thuật toán sàng với  $n = 100$ .

### Bài giảng của Eratosthenes

- Bước 1** Trò hãy viết dãy số từ 1 đến 100 trên bảng đất sét của mình. Trò có đủ 100 số từ 1 đến 100 chưa bị xóa trên bảng.
- Bước 2** Xóa số 1, vì 1 không phải là số nguyên tố cũng không phải là hợp số. 1 là số đặc biệt trong dãy số tự nhiên.
- Bước 3** Tìm số chưa bị xóa tiếp theo. Gọi số đó là  $i$ .
- Bước 4** Nếu  $i > 10$  thì trò dừng thuật toán. Toàn bộ các số không bị xóa trên bảng là các số nguyên tố.
- Nếu  $i \leq 10$  thì trò thực hiện Bước 5.
- Bước 5** Xóa các bội số của  $i$  kể từ  $i^2$  đến 100.
- Bước 6** Lặp lại Bước 3.

Eratosthenes



276–194 trước CN

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Sàng Eratosthenes với  $n = 100$ .

(Các số bị xóa viết trong ô đậm)

### Sieve of Eratosthenes Version 1.

/\*\*\*\*\*



```

    Sieve of Eratosthenes
    Ver. 1: List List all primes below n

    *****/
#include <iostream>
#include <cmath>
#include <bitset>

const int MN = 1000000;

using namespace std;

bitset<MN> p; // sieve

// Sieve of Eratosthenes
// List all primes below n
void Sieve(int n) {
    int s = (int)sqrt(n);
    p.set(); // set p all 1
    p.reset(0);
    p.reset(1);
    // delete all even numbers from 4 to n exclusive
    for (int i = 4; i < n; i += 2)
        p[i] = 0;
    for (int i = 2; i <= s; ++i)
        if (p[i])
            for (int j = i*i; j < n; j += i)
                p[j] = 0;
} // Sieve

void Show(int n, int range = 10) {
    if (n < 2) return;
    int d = 1;
    cout << " " << 2;
    for (int i = 3; i < n; i += 2)
        if (p[i]) {
            cout << " " << i;
            ++d;
            if ((d % range) == 0) cout << endl;
        } // if
} // Show

main() {
    int n = 1000;
    Sieve(n);
    Show(n, 20);

    // -----
    cout << "\n T H E      E N D";
    return 0;
}

```

## Sieve of Eratosthenes Version 2.

```

/*****

```

.....

*Cấu trúc dữ liệu và giải thuật C++ (Tom 1) tr. 49*

### Sieve of Eratosthenes

```
*****/
#include <iostream>
#include <fstream>
#include <cmath>
#include <bitset>

const int MN = 1000000;

using namespace std;

bitset<MN> p; // sieve

// Sieve of Eratosthenes
void Sieve(int n) {
    int s = (int)sqrt(n);
    p.set(); // set p all 1
    p.reset(0);
    p.reset(1);
    // delete all even numbers from 4 to n exclusive
    for (int i = 4; i < n; i += 2)
        p[i] = 0;
    for (int i = 2; i <= s; ++i)
        if (p[i])
            for (int j = i*i; j < n; j += i)
                p[j] = 0;
} // Sieve

void Show(int n, int range = 10) {
    if (n < 2) return;
    int d = 1;
    cout << " " << 2;
    for (int i = 3; i < n; i += 2)
        if (p[i]) {
            cout << " " << i;
            ++d;
            if ((d % range) == 0) cout << endl;
        } // if
} // Show

// Save to file fn
void Save(int n, const char *fn) {
    ofstream f(fn);
    f << 2 << endl;
    for (int i = 3; i < n; i += 2)
        if (p[i])
            f << i << endl;
    f.close();
} // Save

main() {
    int n = 1000;
    Sieve(n);
    Save(n, "Plm.DAT");

    // -----
```

```
cout << "\n T H E      E N D";  
return 0;  
}
```

## 10001st prime (Problem 7 Project Euler)

By listing the first six prime numbers: 2, 3, 5, 7, 11, and 13, we can see that the 6<sup>th</sup> prime is 13.

What is the 10001<sup>st</sup> prime number?

### 1001<sup>st</sup> prime. Version 1 (Brute force)

```

/*****
    10001st prime. Ver. 1
    Brute force
    Answer = 104743
    *****/
#include <iostream>
#include <cmath>
using namespace std;

bool IsPrime(int n) {
    int sn = int(sqrt(n));
    for (int i = 2; i <= sn; ++i)
        if ((n % i) == 0) return false;
    return true;
} // IsPrime

int Ver1() {
    int d = 1; // 2 is the first prime
    for (int n = 3; true; n += 2)
        if (IsPrime(n)) {
            ++d;
            if (d == 10001) return n;
        }
} // Ver1

main() {
    cout << "    " << Ver1();
    // -----
    cout << "\n T H E      E N D";
    return 0;
}

```

### 1001<sup>st</sup> prime. Version 2 (Using Sieve of Eratosthenes)

```

/*****
    10001st prime. Ver. 2
    Using Sieve of Eratosthenes
    Answer = 104743
    P(n)*log(n) ~ n
    *****/

```

```

log(n) = so chu so trong dang nhi phan
log(n) = 10 -> n = 1024
10001*log(n) = 10001*12 = 120000 ~ n

*****/
#include <iostream>
#include <cmath>
#include <bitset>
using namespace std;

const int MAXN = 120000;

bitset<MAXN> p; // sieve

// Sieve of Eratosthenes
void Sieve(int n) {
    int s = (int)sqrt(n);
    p.set(); // set p all 1
    p.reset(0);
    p.reset(1);
    // delete all even numbers from 4 to n exclusive
    for (int i = 4; i < n; i += 2)
        p[i] = 0;
    for (int i = 2; i <= s; ++i)
        if (p[i])
            for (int j = i*i; j < n; j += i)
                p[j] = 0;
} // Sieve

int Ver2() {
    Sieve(MAXN);
    int d = 1; // for 2
    for (int n = 3; n < MAXN; n += 2)
        if (p[n]) {
            ++d;
            if (d == 10001) return n;
        }
} // Ver2

main() {
    cout << " " << Ver2();
    // -----
    cout << "\n T H E      E N D";
    return 0;
}

```

### 1001<sup>st</sup> prime. Version 3

```

/*****

```

```

10001st prime. Ver. 3
Extend primes
Answer = 104743
*****/
#include <iostream>

const int MN = 10002;

using namespace std;

// first 10 primes
int p[MN] = {0, 2, 3, 5, 7, 11, 13, 17, 19, 23, 29};

// x is prime iff x has no factor p[2]..p[n]
// Since x is odd we do not test if a[1] = 2 divides x.
bool Prime(int n, int x) {
    for (int i = 2; i < n; ++i)
        if ((x % p[i]) == 0) return false;
    return true;
}

int Ver3() {
    int x, n, k, m;
    k = 10; // p[k] is the last prime in started sequence
    n = 3; // p[n]=5, p[n+1]=p[4]=7, p[n+1]*p[n+1] > p[k]
    m = p[k];
    while (true) {
        x = m + 2;
        ++n;
        m = p[n]*p[n];
        // Scan from p[k]+2 to m
        for (; x < m; x += 2)
            if (Prime(n, x)) {
                p[++k] = x;
                if (k == 10001) return x;
            } // if
    } // while
} // Ver3

main() {
    cout << "      " << Ver3();
    // -----
    cout << "\n T H E      E N D";
    return 0;
}

```

## Largest prime factor (Problem 3 Project Euler)

The prime factors of 13195 are 5, 7, 13 and 29.

What is the largest prime factor of the number 600851475143 ?

Đặt tên là một nghệ thuật

### Largest prime factor: version 1

```
/******
Largest prime factor of 600851475143?
Ver. 1 Answer = 6857
*****/
#include <iostream>
#include <math.h>

using namespace std;

typedef unsigned long long Long; // size = 8 bytes = 64 bits

Long N = 600851475143; // sqrt = 775146

int V1(Long n) {
    int s = int(sqrt((double) n));
    int p, pmax;
    p = 2; // case: n is an even number
    if ((n % p) == 0) {
        pmax = p;
        while ((n % p) == 0) n /= p;
    }
    // n is an odd number
    for (p = 3; (n > 1) && (p <= s); p += 2) {
        if ((n % p) == 0) {
            pmax = p;
            while ((n % p) == 0) n /= p;
        } // if
    } // for
    return (n > 1) ? n : pmax;
} // V1

main() {
    cout << V1(N);
    cout << "\n T H E      E N D";
    return 0;
}
```

### Largest prime factor: version 2

```

/*****
    Largest prime factor of 600851475143?
    Ver. 2 Answer = 6857
    Ver. 2
    Using Sieve of Eratosthenes
    *****/
#include <iostream>
#include <cmath>
#include <bitset>
using namespace std;

typedef long long Long;
const Long N = 600851475143;

const int MAXN = 1000000;

bitset<MAXN> p; // sieve

// Sieve of Eratosthenes
void Sieve(int n) {
    int s = (int)sqrt(n);
    p.set(); // set p all 1
    p.reset(0);
    p.reset(1);
    // delete all even numbers from 4 to n exclusive
    for (int i = 4; i < n; i += 2)
        p[i] = 0;
    for (int i = 2; i <= s; ++i)
        if (p[i])
            for (int j = i*i; j < n; j += i)
                p[j] = 0;
} // Sieve

Long Factor(Long n, Long sn) {
    for (int x = 3; x <= sn; x += 2) {
        if (p[x]) { // x is a prime
            if ((n % x) == 0) {
                Long a = n/x; // n = a*x
                n /= x;
                return (a > x) ? a : x;
            } // if n
        } // if p
    } // for x
    return n;
} // Factor

int Ver2() {
    Long n = N;
    int sn = int(sqrt((float)n)) + 1;
    Sieve(sn);
    while (1) {
        Long m = Factor(n, sn);
        if (n == m)
            return m;
        else n = m;
        sn = int(sqrt((float)n)) + 1;
    }
}

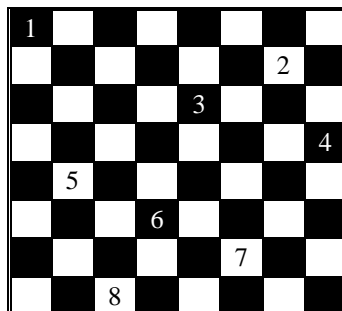
```



```
    } // while
} // Ver2

main() {
    cout << "  " << Ver2();
    // -----
    cout << "\n T H E      E N D";
    return 0;
}
```

## Queens



Tư duy trong sáng, minh bạch

### Queens Version 1. Tìm một nghiệm

```
/******  
    Queens. Ver. 1: Tim mot nghiem  
*****  
#include <iostream>  
#include <cmath>  
  
using namespace std;  
  
const int MN = 100;  
int v[MN]; // v[k] = dong dat Hau k  
  
void Answer(int n) {  
    for (int i = 1; i <= n; ++i)  
        cout << " " << v[i];  
}  
  
// Neu Hau k dung tren dong i  
// thi ko dung voi cac Hau 1..k-1  
bool NicePlace(int n, int k, int i) {  
    for (int j = 1; j < k; ++j)  
        if (v[j] == i || k-j == abs(v[j]-i))  
            return false;  
    return true;  
} // NicePlace  
  
// Di chuyen Hau k tu dong v[k]+1  
// den dong xuong n, tim dong dau tien
```

```

// không dùng do voi cac Hau dat truoc
// 1..k-1
int Find(int n, int k) {
    for (int i = v[k]+1; i <= n; ++i)
        if (NicePlace(n, k, i)) return i;
    return 0;
} // Find

// Back Tracking
void Queens(int n) {
    cout << "\n\n " << n << " Queen(s): ";
    for (int i = 1; i <= n; ++i) v[i] = 0;
    int k = 1; // Cam Hau 1

    while (1) {
        if (k > n) {
            Answer(n);
            return;
        }
        if (k < 1) {
            cout << "\n No solution.";
            return;
        }
        v[k] = Find(n,k);
        if (v[k] > 0) ++k;
        else --k;
    } // while
} // Queens

main() {
    for (int n = 1; n <= 20; ++n)
        Queens(n);
    cout << "\n T H E      E N D ";
    return 0;
}

```

## Queens Vesion 2. Tìm mọi nghiệm

Thay đổi ít nhất nhưng hưởng lợi nhiều nhất

```

/*****
    Queens. Ver 2: All solutions
    *****/
#include <iostream>
#include <cmath>

using namespace std;

```

```

const int MN = 100;
int v[MN]; // v[k] = dong dat Hau k

// Hien thi nghiem thu d
void Answer(int n, int d = 1) {
    cout << "\n " << d << ". ";
    for (int i = 1; i <= n; ++i)
        cout << " " << v[i];
}

// Neu Hau k dung tren dong i
// thi ko dung voi cac Hau 1..k-1
bool NicePlace(int n, int k, int i) {
    for (int j = 1; j < k; ++j)
        if (v[j] == i || k-j == abs(v[j]-i))
            return false;
    return true;
} // NicePlace

// Di chuyen Hau k tu dong v[k]+1
// den dong xuong n, tim dong dau tien
// khong dung do voi cac Hau dat truoc
// 1..k-1
int Find(int n, int k) {
    for (int i = v[k]+1; i <= n; ++i)
        if (NicePlace(n, k, i)) return i;
    return 0;
} // Find

// Back Tracking
void Queens(int n) {
    cout << "\n\n " << n << " Queen(s): ";
    for (int i = 1; i <= n; ++i) v[i] = 0;
    int k = 1; // Cam Hau 1

    while (1) {
        if (k > n) {
            Answer(n);
            return;
        }
        if (k < 1) {
            cout << "\n No solution.";
            return;
        }
        v[k] = Find(n,k);
        if (v[k] > 0) ++k;
        else --k;
    } // while
} // Queens

// Back Tracking
void QueensAll(int n) {
    int d = 0; // dem so nghiem
    cout << "\n\n " << n << " Queen(s): ";
    for (int i = 1; i <= n; ++i) v[i] = 0;
    int k = 1; // Cam Hau 1

```

```

while (1) {
    if (k > n) {
        ++d;
        Answer(n,d);
        k = n; // gia sai
    }
    if (k < 1) {
        if (d > 0)
            cout << "\n Total " << d << " solution.";
        else cout << "\n no " << " solution.";
        return;
    }
    v[k] = Find(n,k);
    if (v[k] > 0) ++k;
    else --k;
} // while
} // Queens

main() {
    Queens(8);
    QueensAll(8);
    cout << "\n T H E    E N D ";
    return 0;
}

```

### Queens Vesion 3. Hai trong một

Hà Nội không ... được đâu !

```

/*****
    Queens. Ver 3: Hai trong mot
*****/

#include <iostream>
#include <cmath>

using namespace std;

const int MN = 100;
int v[MN]; // v[k] = dong dat Hau k

// Hien thi nghiem thu d
void Answer(int n, int d = 1) {
    cout << "\n " << d << ". ";
    for (int i = 1; i <= n; ++i)
        cout << " " << v[i];
}

```

```

// Neu Hau k dung tren dong i
// thi ko dung voi cac Hau 1..k-1
bool NicePlace(int n, int k, int i) {
    for (int j = 1; j < k; ++j)
        if (v[j] == i || k-j == abs(v[j]-i))
            return false;
    return true;
} // NicePlace

// Di chuyen Hau k tu dong v[k]+1
// den dong xuong n, tim dong dau tien
// khong dung do voi cac Hau dat truoc
// 1..k-1
int Find(int n, int k) {
    for (int i = v[k]+1; i <= n; ++i)
        if (NicePlace(n, k, i)) return i;
    return 0;
} // Find

// Back Tracking
void Queens(int n, int songhiem = 1) {
    int d = 0; // dem so nghiem
    cout << "\n\n " << n << " Queen(s): ";
    for (int i = 1; i <= n; ++i) v[i] = 0;
    int k = 1; // Cam Hau 1

    while (1) {
        if (k > n) {
            ++d;
            Answer(n,d);
            if (songhiem == 1) return;
            k = n; // gia sai
        }
        if (k < 1) {
            cout << "\n Total " << d << " solution.";
            return;
        }
        v[k] = Find(n,k);
        if (v[k] > 0) ++k;
        else --k;
    } // while
} // Queens

main() {
    // Queens(3);
    Queens(4,2);
    cout << "\n T H E   E N D ";
    return 0;
}

```

## Rhythm

### Auto Test

#### E5 and data generation

Given  $n$  numbers  $a[0..n-1]$ . Write a function named Rhythm giving the following values:

- 1 if all the numbers are equal:  $a[0] = a[1] = \dots = a[n-1]$
  - 2 if  $a[0..n-1]$  is a strong ascending sequence:  $a[0] < a[1] < \dots < a[n-1]$
  - 3 if  $a[0..n-1]$  is a ascending sequence:  $a[0] \leq a[1] \leq \dots \leq a[n-1]$
  - 4 if  $a[0..n-1]$  is a strong descending sequence:  $a[0] > a[1] > \dots > a[n-1]$
  - 5 if  $a[0..n-1]$  is a ascending sequence:  $a[0] \geq a[1] \geq \dots \geq a[n-1]$
- 0 or else.

E5									
a	0	1	2	3	4	5	6	7	8
1									
2									
3									
4									
5									
0									
?									

### Rhythm Version 1

```
/******  
    Rhythm.cpp Ver. 1  
    1  a[0] = a[1] = ... = a[n-1]  
    2  a[0] < a[1] < ... < a[n-1]  
    3  a[0]  a[1]    ...  a[n-1]  
    4  a[0] > a[1] > ... > a[n-1]  
    5  a[0]  a[1]    ...  a[n-1]  
    0 or else.  
*****/
```

```

#include <iostream>
using namespace std;

int a[] = { 3,13,33,33,37,
           39,130,130,133,133,
           300,300,300,300,300};

int Rhythm(int a[], int n) {
    int g, t, b, r;
    g = t = b = 0;
    for (int i = 1; i < n; ++i) {
        if (a[i] == a[i-1]) ++b;
        else if (a[i] > a[i-1]) ++t;
        else // a[i] < a[i-1]
            ++g;
    }

    if (b > 0) b = 1;
    if (t > 0) t = 1;
    if (g > 0) g = 1;
    r = 4*g + 2*t + b;
    if (r > 5) r = 0;
    return r;
} // Rhythm

main() {
    int n = 15;
    cout << Rhythm(a, n); // 3
    //-----
    // cout << endl << " T H E   E N D . ";
    return 0;
}

```

## Rhythm Version 2

```

/*****
Rhythm.cpp Ver. 2
1  a[0] = a[1] = ... = a[n-1]
2  a[0] < a[1] < ... < a[n-1]
3  a[0]  a[1]    ...  a[n-1]
4  a[0] > a[1] > ... > a[n-1]
5  a[0]  a[1]    ...  a[n-1]
0 or else.
*****/

#include <iostream>
#include <ctime>
#include <windows.h>

using namespace std;

const int MN = 1000000; // 1M
int a[MN] = { 3,13,33,33,37,

```



```

        39,130,130,133,133,
        300,300,300,300,300};

int Gen1(int n) { // equal
    int v = rand() % 100;
    for (int i = 0; i < n; ++i)
        a[i] = v;
    return n;
} // Gen1

int Gen2(int n) { // strong asc
    a[0] = rand() % 5;
    for (int i = 1; i < n; ++i)
        a[i] = a[i-1] + ((rand() % 5) + 1);
    return n;
} // Gen2

int Gen3(int n) { // asc
    a[0] = rand() % 5;
    for (int i = 1; i < n; ++i)
        a[i] = a[i-1] + (rand() % 5);
    return n;
} // Gen3

int Gen0(int n) { // asc
    Gen3(n);
    a[rand() % n] = rand() % n;
    return n;
} // Gen0

/*-----
g t b r
0 0 0 ?
0 0 1 1
0 1 0 2
0 1 1 3
1 0 0 4
1 0 1 5
1 1 0 0 (6)
1 1 1 0 (7)
-----*/
int Rhythm(int a[], int n) {
    int g, t, b, r;
    g = t = b = 0;
    for (int i = 1; i < n; ++i) {
        if (a[i] == a[i-1]) b = 1;
        else if (a[i] > a[i-1]) t = 1;
        else // a[i] < a[i-1]
            g = 1;
    }
    return (4*g + 2*t + b) % 7 % 6;
} // Rhythm

// Doi cho cac phan tu cach deu dau va cuoi
void Rev(int a[], int n) {
    int d = 0, c = n-1, x;
    while(d < c) {

```

```

        // doi cho a[d] a[c]
        x = a[d];
        a[d] = a[c];
        a[c] = x;
        ++d; --c;
    } // while
} // Rev

void Test2 () {
    srand(time(NULL));
    int n = Gen0(100000);
    cout << "\n " << Rhythm(a, n); // 3
    Rev(a, n);
    cout << "\n " << Rhythm(a, n); // 5
} // Test2

void Test1 () {
    int n = 15;
    cout << "\n " << Rhythm(a, n); // 3
    Rev(a, n);
    cout << "\n " << Rhythm(a, n); // 5
} // Test1

main() {
    Test2();
    //-----
    // cout << endl << " T H E   E N D . ";
    return 0;
}

```

## Range Check Error

Range Check Error là lỗi thường gặp  
Biết điều khiển index sẽ tăng hiệu quả chương trình  
"Anh cố tìm điều dễ ghét trong em  
Tìm được rồi anh càng thấy yêu thêm"  
Việt Phương

$p$  is a permutations of  $n$  elements  $1, 2, \dots, n$ . If  $x$  and  $y$  in  $p$ ,  $x < y$  and  $x$  stand after  $y$  then we call  $(x, y)$  is a reversed pair in  $p$ . Find the longest reversed pair in  $p$ .

### Example

$p = (\underline{4}, 6, 1, 5, 9, 2, 7, \underline{3}, 8)$

$p = (3, 5, \underline{8}, 9, 1, 2, 4, 6, \underline{7})$ ;

```
/******  
longest reversed pair in a permutation  
*****/  
#include <iostream>  
#include <ctime>  
#include <windows.h>  
using namespace std;  
  
const int MN = 1000000; // 1M  
  
// int p[MN+1] = {4, 6, 1, 5, 9, 2, 7, 3, 8}; // (0,7)  
int p[MN+1] = {3, 5, 8, 9, 1, 2, 4, 6, 7}; // (2,8)  
  
int Gen(int n) {  
    for (int i = 0; i < n; ++i)  
        p[i] = i+1; // 1, 2, .., n  
    int a = 10, b = n-a;  
    int nn = n+n;  
    for (int i = 0; i < nn; ++i) {  
        int u = a + (rand() % (b-a)); // so ngau nhien a..b  
        int v = a + (rand() % (b-a)); // so ngau nhien a..b  
        int t = p[u];  
        p[u] = p[v];  
        p[v] = t;  
    }  
    return n;  
} // Gen  
  
void LRP(int n) {  
    int imax = 0, jmax = 0;  
    for (int i = 0; i < n; ++i) {
```

```

        for (int j = n-1; j > i; --j) {
            if (p[i] > p[j]) {
                if (j-i > jmax-imax) {
                    imax = i;
                    jmax = j;
                } // if i
            } // if p
        } // for j
    } // for i
    cout << "\n " << imax << " : " << jmax;
} // LRP

void NewLRP(int n) {
    int imax = 0, d = 0;
    // -----[i]*****[i+d]-----*****[n-1], d = *****
    for (int i = 0; i < n-d; ++i) {
        for (int j = n-1; j > i+d; --j) {
            if (p[i] > p[j]) {
                imax = i;
                d = j - i;
                break; // j
            } // if p
        } // for j
    } // for i
    cout << "\n " << imax << " : " << imax+d;
} // LRP1

void Test() {
    int n = Gen(10000); // 100000;
    int t1 = time(NULL);
    NewLRP(n);
    int t2 = time(NULL);
    cout << "\n Time = " << difftime(t2, t1);
    LRP(n);
    int t3 = time(NULL);
    cout << "\n Time = " << difftime(t3, t2);
} // Test

main() {
    srand(time(NULL));
    Test();
    // -----
    cout << "\n T H E      E N D";
    return 0;
}

```

## New RomanNumerals

"Anh cố tìm điều dễ ghét trong em  
Tìm được rồi anh càng thấy yêu thêm"  
Việt Phương

```
/******  
    New Roman Numerals System  
    N = 0  
    (s) = 1000*s  
    (VI) = 6000  
*****/  
#include <iostream>  
#include <cmath>  
using namespace std;  
  
typedef long long Long;  
  
int ival[] =  
{1000, 900, 500, 400, 100, 90, 50, 40, 10, 9, 5, 4, 1};  
  
string rval[] =  
{ "M", "CM", "D", "CD", "C", "XC", "L", "XL", "X", "IX", "V", "IV", "I"};  
int len = 13;  
  
const int MN = 90;  
int f[MN+1];  
  
void Go() {  
    cout << " ? ";  
    fflush(stdin);  
    if (cin.get() == '.')  
        exit(0);  
} // Go  
  
string Str(char c) {  
    string s = "";  
    s += c;  
    return s;  
} // Str  
  
// Vi tri xuat hien string r trong rval  
int Rpos(string r) {  
    for (int i = 0; i < len; ++i)  
        if (rval[i] == r) return i;  
    return -1;  
} // Rpos
```

```

// Vi tri xuat hien int v trong ival
int Ipos(int v) {
    for (int i = 0; i < len; ++i)
        if (ival[i] == v) return i;
    return -1;
} // Ipos

// rom r -> Long
Long RomInt(string r) {
    int i = Rpos(r);
    return (i < 0) ? 0 : ival[i];
} // RomInt

// Long v -> rom
string IntRom(Long v) {
    int i = Ipos(v);
    return (i < 0) ? "" : rval[i];
} // IntRom

// CM = 900, CD = 40; XC = 90, XL = 40; IX = 9, IV = 4
int Pair(char c, char cc, int &d) {
    string s = Str(c);
    switch(c) {
        case 'C': // CM | CD
            if (cc == 'M' || cc == 'D') s += cc;
            break;
        case 'X': // XC | XL
            if (cc == 'C' || cc == 'L') s += cc;
            break;
        case 'I': // IX | IV
            if (cc == 'X' || cc == 'V') s += cc;
            break;
    } // switch
    d = s.length();
    return RomInt(s);
} // Pair

Long ToInt(string inpr) { // convert rom to int CDLVI -> 456
    // Bo cac dau cach: inpr -> r
    string r = "";
    for (int i = 0; i < inpr.length(); ++i)
        if (inpr[i] != 32) r += inpr[i];
    // if (r == "") return 0;
    r += "#"; // Them linh canh cuoi xau: guards
    Long val = 0; // output value
    int d = 0;
    for (int i = 0; i < r.length(); i += d) {
        val += Pair(r[i], r[i+1], d);
    } // for
    return val;
} // ToInt

// s[i..j]
string Segment(string s, int d, int c) {
    string w = "";
    for (int i = d; i <= c; ++i)

```

```

        w += s[i];
    return w;
} // Segment

Long ToIntNew(string inpr) { // convert rom to int CDLVI -> 456
    // Bo cac dau cach: inpr -> r
    string r = "";
    for (int i = 0; i < inpr.length(); ++i)
        if (inpr[i] != 32) r += inpr[i];
    if (r == "") return 0;
    if (r == "N") return 0;
    int dm = 0; // dem so chu M
    int k = 0;
    Long val = 0;
    int n = r.length();
    if (r[0] == '(') {
        for (int i = 1; i < n; ++i) {
            if (r[i] == ')') {
                k = i;
                break;
            }
        } // for i
        if (k == 0) {
            cerr << "\n Syntax error.";
            exit(0);
        } // if k
    } // (
    return (k > 0)
        ? ToInt(Segment(r, 1, k)) * 1000 + ToInt(Segment(r, k + 1, n - 1))
        : ToInt(r);
} // ToIntNew

string ToRom(Long n) { // convert int to rom: 456 -> CDLVI
    string r = "";
    for (int i = 0; i < len; ++i) {
        while (n >= ival[i]) {
            r += IntRom(ival[i]);
            n -= ival[i];
        } // while
    } // for
    return (r == "") ? "N" : r;
} // ToRom

string ToRomNew(Long n) { // convert int to rom: 456 -> CDLVI
    if (n < 5000) return ToRom(n);
    Long dm = n / 1000; // so luong 1000
    n %= 1000; // du
    string r = "(" + ToRom(dm) + ")";
    if (n > 0) r += ToRom(n);
    return r;
} // ToRomNew

void Test() {
    // Long n = 40000; // (XL)
    // Long n = 43268; // (XL) CCLXVIII
    // cout << n << " = " << ToRomNew(n);
    Long n = 60036;

```

```

    string r = ToRomNew(n);
    Long v = ToIntNew(" (LX)XXXVI ");
    cout << n << " " << r << " " << v << " " << ToRomNew(v);

}

main() {
    Test();
    cout << "\n T H E    E N D";
    return 0;
}

```



## New Roman Fibonacci (R+)

```

/*****
    New Roman Fibonacci
    Nice Max = Fib(31)

*****/
#include <iostream>
#include <cmath>
using namespace std;

typedef long long Long;

/*****
    New Roman Numerals System
    N = 0
    (s) = 1000*s
    (VI) = 6000
*****/
#include <iostream>
#include <cmath>
using namespace std;

typedef long long Long;

int ival[] =
{1000, 900, 500, 400,100, 90, 50, 40, 10, 9, 5, 4, 1};

string rval[]=
{"M", "CM","D", "CD","C", "XC", "L", "XL", "X", "IX", "V", "IV", "I"};
int len = 13;

const int MN = 90;
Long f[MN+1];

void Go() {
    cout << " ? ";
    fflush(stdin);
    if (cin.get() == '.')
        exit(0);
} // Go

string Str(char c) {
    string s = "";
    s += c;
    return s;
} // Str

// Vị trí xuất hiện string r trong rval
int Rpos(string r) {
    for (int i = 0; i < len; ++i)
        if (rval[i] == r) return i;
    return -1;
} // Rpos

```

```

// Vi tri xuat hien int v trong ival
int Ipos(int v) {
    for (int i = 0; i < len; ++i)
        if (ival[i] == v) return i;
    return -1;
} // Ipos

// rom r -> Long
Long RomInt(string r) {
    int i = Rpos(r);
    return (i < 0) ? 0 : ival[i];
} // RomInt

// Long v -> rom
string IntRom(Long v) {
    int i = Ipos(v);
    return (i < 0) ? "" : rval[i];
} // IntRom

// CM = 900, CD = 40; XC = 90, XL = 40; IX = 9, IV = 4
int Pair(char c, char cc, int &d) {
    string s = Str(c);
    switch(c) {
        case 'C': // CM | CD
            if (cc == 'M' || cc == 'D') s += cc;
            break;
        case 'X': // XC | XL
            if (cc == 'C' || cc == 'L') s += cc;
            break;
        case 'I': // IX | IV
            if (cc == 'X' || cc == 'V') s += cc;
            break;
    } // switch
    d = s.length();
    return RomInt(s);
} // Pair

Long ToInt(string inpr) { // convert rom to int CDLVI -> 456
    // Bo cac dau cach: inpr -> r
    string r = "";
    for (int i = 0; i < inpr.length(); ++i)
        if (inpr[i] != 32) r += inpr[i];
    // if (r == "") return 0;
    r += "#"; // Them linh canh cuoi xau: guards
    Long val = 0; // output value
    int d = 0;
    for (int i = 0; i < r.length(); i += d) {
        val += Pair(r[i], r[i+1], d);
    } // for
    return val;
} // ToInt

// s[i..j]
string Segment(string s, int d, int c) {
    string w = "";
    for (int i = d; i <= c; ++i)

```

```

        w += s[i];
    return w;
} // Segment

Long ToIntNew(string inpr) { // convert rom to int CDLVI -> 456
    // Bo cac dau cach: inpr -> r
    string r = "";
    for (int i = 0; i < inpr.length(); ++i)
        if (inpr[i] != 32) r += inpr[i];
    if (r == "") return 0;
    if (r == "N") return 0;
    int dm = 0; // dem so chu M
    int k = 0;
    Long val = 0;
    int n = r.length();
    if (r[0] == '(') {
        for (int i = 1; i < n; ++i) {
            if (r[i] == ')') {
                k = i;
                break;
            }
        } // for i
        if (k == 0) {
            cerr << "\n Syntax error.";
            exit(0);
        } // if k
    } // (
    return (k > 0)
        ? ToInt(Segment(r, 1, k)) * 1000 + ToInt(Segment(r, k + 1, n - 1))
        : ToInt(r);
} // ToIntNew

string ToRom(Long n) { // convert int to rom: 456 -> CDLVI
    string r = "";
    for (int i = 0; i < len; ++i) {
        while (n >= ival[i]) {
            r += IntRom(ival[i]);
            n -= ival[i];
        } // while
    } // for
    return (r == "") ? "N" : r;
} // ToRom

string ToRomNew(Long n) { // convert int to rom: 456 -> CDLVI
    if (n < 5000) return ToRom(n);
    Long dm = n / 1000; // so luong 1000
    n %= 1000; // du
    string r = "(" + ToRom(dm) + ")";
    if (n > 0) r += ToRom(n);
    return r;
} // ToRomNew

/*-----
                        Fibonacci
-----*/

int AllFib() {

```

```

Long a, b, c;
a = b = 1;
f[1] = f[2] = 1;
for (int i = 3; i <= MN; ++i) {
    c = a + b;
    f[i] = c;
    a = b;
    b = c;
} // for
} // AllFib

Long Fib(int n) {
    return (n <= MN) ? f[n] : 0;
} // Fib

string RFib(string r) {
    int i = ToIntNew(r);
    return (i <= MN) ? ToRomNew(f[i]) : "";
} // RFib

void Test() {
    cout << "\n ----- T E S T -----";
    int nn = 20;
    for (int n = 1; n <= nn; ++n)
        cout << "\n n = " << n << "   Fib = "
            << Fib(n) << "   RFib = " << RFib(ToRomNew(n));
} // Test

void Test1() {
    cout << "\n ----- T E S T 1 -----";
    int nn = 31; // 40; // 90; // 25; // 90
    AllFib();
    for (int n = 1; n <= nn; ++n) {
        cout << "\n n = " << n << "   Fib = "
            << Fib(n) << "   RFib = " << RFib(ToRomNew(n));
        // Go();
    } // for
} // Test1

main() {
    Test();
    Test1();
    cout << "\n T H E   E N D";
    return 0;
}

```

## New Roman Fibonacci Again (R++)

```

/*****
    New Roman Fibonacci (R++)

*****/
#include <iostream>
#include <cmath>
using namespace std;

typedef long long Long;

/*****
    New Roman Numerals System
    N = 0
    (s) = 1000*s
    (VI) = 6000
    ((VI)) = 6000.000
*****/
#include <iostream>
#include <cmath>
using namespace std;

typedef long long Long;

int ival[] =
{1000, 900, 500, 400,100, 90, 50, 40, 10, 9, 5, 4, 1};

string rval[] =
{"M", "CM","D", "CD","C", "XC", "L", "XL", "X", "IX", "V", "IV", "I"};
int len = 13;

const int MN = 90;
Long f[MN+1];

void Go() {
    cout << " ? ";
    fflush(stdin);
    if (cin.get() == '.')
        exit(0);
} // Go

string Str(char c) {
    string s = "";
    s += c;
    return s;
} // Str

// Vi tri xuat hien string r trong rval
int Rpos(string r) {
    for (int i = 0; i < len; ++i)
        if (rval[i] == r) return i;
}

```

```

    return -1;
} // Rpos

// Vi tri xuat hien int v trong ival
int Ipos(int v) {
    for (int i = 0; i < len; ++i)
        if (ival[i] == v) return i;
    return -1;
} // Ipos

// rom r -> Long
Long RomInt(string r) {
    int i = Rpos(r);
    return (i < 0) ? 0 : ival[i];
} // RomInt

// Long v -> rom
string IntRom(Long v) {
    int i = Ipos(v);
    return (i < 0) ? "" : rval[i];
} // IntRom

// CM = 900, CD = 40; XC = 90, XL = 40; IX = 9, IV = 4
int Pair(char c, char cc, int &d) {
    string s = Str(c);
    switch(c) {
        case 'C': // CM | CD
            if (cc == 'M' || cc == 'D') s += cc;
            break;
        case 'X': // XC | XL
            if (cc == 'C' || cc == 'L') s += cc;
            break;
        case 'I': // IX | IV
            if (cc == 'X' || cc == 'V') s += cc;
            break;
    } // switch
    d = s.length();
    return RomInt(s);
} // Pair

Long ToInt(string inpr) { // convert rom to int CDLVI -> 456
    // Bo cac dau cach: inpr -> r
    string r = "";
    for (int i = 0; i < inpr.length(); ++i)
        if (inpr[i] != 32) r += inpr[i];
    // if (r == "") return 0;
    r += "#"; // Them linh canh cuoi xau: guards
    Long val = 0; // output value
    int d = 0;
    for (int i = 0; i < r.length(); i += d) {
        val += Pair(r[i], r[i+1], d);
    } // for
    return val;
} // ToInt

// s[i..j]
string Segment(string s, int d, int c) {

```

```

    string w = "";
    for (int i = d; i <= c; ++i)
        w += s[i];
    return w;
} // Segment

Long ToIntPP(string inpr) { // convert rom to int CDLVI -> 456
    // Bo cac dau cach: inpr -> r
    string r = "";
    for (int i = 0; i < inpr.length(); ++i)
        if (inpr[i] != 32) r += inpr[i];
    if (r == "") return 0;
    if (r == "N") return 0;
    int mn = 0; // dem so lan (
    int d = 0; // so ki tu Roman
    int deg = 1; // bac can nhan them
    Long val = 0;
    int n = r.length();
    int a, b; // a...b
    for (int i = 0; i < n; ++i) {
        if (r[i] == '(') {
            ++mn;
            d = 0;
            a = i;
            deg *= 1000;
            if (deg <= 0) {
                cerr << "\n Many (. ";
                exit(0);
            }
        }
        else if (r[i] == ')') {
            --mn;
            if (mn == 0) {
                b = i; // (([a:(]...)) [b:])
                val += ToInt(Segment(r,a+1,a+d))*deg;
                // mn = 0
                d = 0;
                deg = 1;
            }
        }
        else ++d;
    } // for i

    if (mn != 0) {
        cerr << "\n Syntax Error.";
        exit(0);
    }

    if (val == 0) // ko gap ()
        return ToInt(r);
    // Gap () hoac ()): ((( [b:])
    return val + ToInt(Segment(r,b+1,n-1));
} // ToIntPP

string ToRom(Long n) { // convert int to rom: 456 -> CDLVI
    string r = "";
    for (int i = 0; i < len; ++i) {

```

```

        while (n >= ival[i]) {
            r += IntRom(ival[i]);
            n -= ival[i];
        } // while
    } // for
    return r;
} // ToRom

string ToRomPP(Long n) { // convert int to rom: 456 -> CDLVI
    if (n == 0) return "N";
    if (n <= 3000) return ToRom(n);
    string r = "";
    // n = 123456789 -> ((CXXIII)) (CDLVI) DCCLXXXIX
    Long res;
    string mn = "", dn = "";
    while (n > 0) {
        res = n % 1000;
        n /= 1000;
        if (res != 0)
            r = mn + ToRom(res) + dn + r;
        // cout << "\n res = " << res << " -> " << r;
        mn += "(";
        dn += ")";
    }
    return r;
} // ToRomPP

/*-----
                        Fibonacci
-----*/

int AllFib() {
    Long a, b, c;
    a = b = 1;
    f[1] = f[2] = 1;
    for (int i = 3; i <= MN; ++i) {
        c = a + b;
        f[i] = c;
        a = b;
        b = c;
    } // for
} // AllFib

Long Fib(int n) {
    return (n <= MN) ? f[n] : 0;
} // Fib

string RFib(string r) {
    int i = ToIntPP(r);
    return (i <= MN) ? ToRomPP(f[i]) : "";
} // RFib

void Test() {
    cout << "\n ----- T E S T -----";
    AllFib();
    int nn = 90; // 20;
    for (int n = 1; n <= nn; ++n) {

```



```

        cout << "\n n = " << n << "   Fib = "
        << Fib(n) << "   RFib = " << RFib(ToRomPP(n));
        Go();
    }
} // Test

main() {
    Test();
    cout << "\n " << ToRomPP(3000001);
    cout << "\n T H E   E N D";
    return 0;
}

main() {
    // Test();
    cout << ToRomPP(2000000);
    cout << "\n T H E   E N D";
    return 0;
}

```

August 6 2020