Bài thực hành 2: Hàm và tối ưu mã nguồn

Phần 1. Thực hành về hàm

1.1 Truyền tham trị, tham chiếu và tham số ngầm định

Bài tập 1: Truyền tham trị

Viết hàm tính độ dài cạnh huyền của tam giác theo độ hai cạnh góc vuông.

In [ ]:

```
#include <stdio.h>
#include <math.h>

float get_hypotenuse(float x, float y) {
    /*****************
    # YOUR CODE HERE #
    *****************/
}

int main(){
    float x, y;
    scanf("%f%f", &x, &y);

    float z = get_hypotenuse(x, y);
    printf("z = %.2f\n", z);

    return 0;
}
```

Bài tập 2: Truyền tham chiếu

Viết hàm hoán vị vòng tròn 3 biến a, b, c. Sau khi thực hiện hàm, các biến a, b, c tương ứng nhận các giá trị mới b, c, a.

In [ ]:

```
#include <stdio.h>

void rotate(int &x, int &y, int &z) {
    /*****************
    # YOUR CODE HERE #
    *****************/
}

int main() {
    int x, y, z;

    //# Nhập 3 số nguyên
    /*****************
    # YOUR CODE HERE #
    *****************/
```

```
    printf("Before: %d, %d, %d\n", x, y, z);
    rotate(x, y, z);
    printf("After: %d, %d, %d\n", x, y, z);

    return 0;
}
```

Bài tập 3: Tham số ngầm định

Viết chương trình yêu cầu nhập giá trị cho số nguyên x nhỏ hơn 100. In ra giá trị ax2+bx+cax2+bx+c với a, b, c định sẵn.

In [ ]:

```
#include <stdio.h>

//# Viết hàm get_value
/*****************
# YOUR CODE HERE #
*****************/

int main(){
    int x;
    scanf("%d", &x);

    int a = 2; //# giá trị mặc định của a
    int b = 1; //# giá trị mặc định của b
    int c = 0; //# giá trị mặc định của c

    //# Nhập 3 số nguyên a, b, c từ bàn phím
    /*****************
    # YOUR CODE HERE #
    *****************/

    printf("a=2, b=1, c=0: %d\n", get_value(x));
    printf("a=%d, b=1, c=0: %d\n", a, get_value(x, a));
    printf("a=%d, b=%d, c=0: %d\n", a, b, get_value(x, a, b));
    printf("a=%d, b=%d, c=%d: %d\n", a, b, c, get_value(x, a, b, c));

    return 0;
}
```

1.2 Đa năng hóa hàm

Bài tập 4: Đa năng hóa hàm

Viết các hàm tính lập phương của số nguyên và số thực.

In [ ]:

```
#include <stdio.h>

int cube(int x) {
    //# trả về lập phương của x
    /*****************
    # YOUR CODE HERE #
```

```
    *****************/
}

//# viết hàm tính lập phương của một số kiểu double
/*****************
# YOUR CODE HERE #
*****************/

int main() {
    int n;
    double f;
    scanf("%d %lf", &n, &f);

    printf("Int: %d\n", cube(n));
    printf("Double: %.2lf\n", cube(f));

    return 0;
}
```

Bài tập 5: Đa năng hóa toán tử

Viết các toán tử tính tổng, hiệu, tích và thương của hai số phức.

In [ ]:

```
#include <iostream>
#include <ostream>
#include <math.h>
#include <iomanip>

using namespace std;

struct Complex {
    double real;
    double imag;
};

Complex operator + (Complex a, Complex b) {
    /*****************
    # YOUR CODE HERE #
    *****************/
}

Complex operator - (Complex a, Complex b) {
    /*****************
    # YOUR CODE HERE #
    *****************/
}

Complex operator * (Complex a, Complex b) {
    /*****************
    # YOUR CODE HERE #
```

```
    ****************/
}

Complex operator / (Complex a, Complex b) {
    /*****************
    # YOUR CODE HERE #
    *****************/
}

ostream& operator << (ostream& out, const Complex &a) {
    out << '(' << std::setprecision(2) << a.real << (a.imag >= 0 ? '+' : '-
') << std::setprecision(2) << fabs(a.imag) << 'i' << ')';
    return out;
}

int main() {
    double real_a, real_b, img_a, img_b;
    cin >> real_a >> img_a;
    cin >> real_b >> img_b;

    Complex a{real_a, img_a};
    Complex b{real_b, img_b};

    cout << a << " + " << b << " = " << a + b << endl;
    cout << a << " - " << b << " = " << a - b << endl;
    cout << a << " * " << b << " = " << a * b << endl;
    cout << a << " / " << b << " = " << a / b << endl;

    return 0;
}
```

1.3 Con trỏ hàm và tham số hóa hàm

Bài tập 6: Con trỏ hàm

Giả thuyết Collatz: bắt đầu từ số dương n bất kỳ, nếu n chẵn thì chia 2, nếu lẻ thì nhân 3 cộng 1, giả thuyết cho rằng ta luôn đi đến $n=1$.
Hãy viết chương trình mô phỏng lại quá trình biến đổi để kiểm chứng giả thuyết với giá trị của n nhập từ bàn phím.

In [ ]:

```
#include <stdio.h>

void print(int n) {
    printf("n=%d\n", n);
}

int mul3plus1(int n) {
    return n * 3 + 1;
}

int div2(int n) {
```

```
    return n / 2;
}

// khai báo các tham số cho các con trỏ hàm odd, even và output
void simulate(int n, /*****************# YOUR CODE HERE #*****************/
)  {
    (*output)(n);
    if (n == 1) return;
    if (n % 2 == 0) {
        n = (*even)(n);
    } else {
        n = (*odd)(n);
    }
    simulate(n, odd, even, output);
}

int main() {
    int (*odd)(int) = NULL;
    int (*even)(int) = NULL;

    /*****************
    # YOUR CODE HERE #
    *****************/

    int n;
    scanf("%d", &n);
    simulate(n, odd, even, print);

    return 0;
}
```

Bài tập 7: Khái quát hóa hàm

Viết hàm tính tổng các phần tử trong hai mảng.
Yêu cầu sử dụng function template để cho phép hàm làm việc với các mảng số nguyên lẫn số thực.

In [ ]:

```
#include <iostream>
using namespace std;

//# viết hàm arr_sum
/*****************
# YOUR CODE HERE #
*****************/

int main() {
    int val;
    cin >> val;

    {
        int a[] = {3, 2, 0, val};
```

```
        int b[] = {5, 6, 1, 2, 7};
        cout << arr_sum(a, 4, b, 5) << endl;
    }
    {
        double a[] = {3.0, 2, 0, val * 1.0};
        double b[] = {5, 6.1, 1, 2.3, 7};
        cout << arr_sum(a, 4, b, 5) << endl;
    }

    return 0;
}
```

1.4 Biểu thức lamda và hàm nặc danh

Bài tập 8: Sắp xếp

Viết hàm so sánh cho thuật toán sắp xếp.

In [ ]:

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <numeric>

using namespace std;

int main() {
    int val1, val2;
    cin >> val1 >> val2;
    vector< vector<int> > a = {
        {1, 3, 7},
        {2, 3, 4, val1},
        {9, 8, 15},
        {10, val2},
    };

    //# sắp xếp các vector trong a theo tổng các phần tử giảm dần
    /*****************
    # YOUR CODE HERE #
    *****************/

    for (const auto &v : a) {
        for (int it : v) {
            cout << it << ' ';
        }
        cout << endl;
    }
    return 0;
}
```

Phần 2. Thực hành về tối ưu mã nguồn

Hãy giải các bài toán sau đây một cách tối ưu nhất có thể, cố gắng sử dụng các kỹ thuật đã được học như inline, static, ...

Bài tập 9: Tính hàm sigmoid

Dưới đây cung cấp đoạn code đơn giản để tính hàm sigmoid theo công thức trực tiếp.
Hãy viết hàm tính xấp xỉ sigmoid(x) đến độ chính xác 10^{-6} và có tốc độ nhanh hơn ít nhất 30% so với code đơn giản.
Gợi ý: sử dụng kỹ thuật "chuẩn bị trước" như trong slide.

In [ ]:

```
#include <vector>
#include <algorithm>
#include <cmath>
#include <ctime>
#include <algorithm>
#include <cstdio>

using namespace std;

const int LIMIT = 100;
const int NUM_ITER = 100000;
const int NUM_INPUTS = NUM_ITER * 100;

double sigmoid_slow(double x) {
    return 1.0 / (1.0 + exp(-x));
}

double x[NUM_INPUTS];

void prepare_input() {
    const int PRECISION = 1000000;
    const double RANGE = LIMIT / 20.0;
    for (int i = 0; i < NUM_INPUTS; ++i) {
        x[i] = RANGE * (rand() % PRECISION - rand() % PRECISION) / PRECISIO
N;
    }
}

//# BEGIN fast code

//# khai báo các biến phụ trợ cần thiết
/*****************
# YOUR CODE HERE #
*****************/

//# hàm chuẩn bị dữ liệu
void precalc() {
    /*****************
    # YOUR CODE HERE #
    *****************/
}
```

```
//# hàm tính sigmoid(x) nhanh sigmoid_fast(x)
inline double sigmoid_fast(double x) {
    /*****************
    # YOUR CODE HERE #
    *****************/
}

//# END fast code

double benchmark(double (*calc)(double), vector<double> &result) {
    const int NUM_TEST = 20;

    double taken = 0;
    result = vector<double>();
    result.reserve(NUM_ITER);

    int input_id = 0;
    clock_t start = clock();
    for (int t = 0; t < NUM_TEST; ++t) {
        double sum = 0;
        for (int i = 0; i < NUM_ITER; ++i) {
            double v = fabs(calc(x[input_id]));
            sum += v;
            if (t == 0) result.push_back(v);
            if ((++input_id) == NUM_INPUTS) input_id = 0;
        }
    }
    clock_t finish = clock();
    taken = (double)(finish - start);
//#  printf("Time: %.9f\n", taken / CLOCKS_PER_SEC);
    return taken;
}

bool is_correct(const vector<double> &a, const vector<double> &b) {
    const double EPS = 1e-6;

    if (a.size() != b.size()) return false;
    for (int i = 0; i < a.size(); ++i) {
        if (fabs(a[i] - b[i]) > EPS) {
            return false;
        }
    }
    return true;
}

int main() {
    prepare_input();
    precalc();

    vector<double> a, b;
```

```
    double slow = benchmark(sigmoid_slow, a);
    double fast = benchmark(sigmoid_fast, b);

    double xval;
    scanf("%lf", &xval);
    printf("%.2f \n", sigmoid_fast(xval));

    if (is_correct(a, b) && (slow/fast > 1.3)) {
        printf("Correct answer! Your code is faster\n");
    } else {
        printf("Wrong answer or your code is not fast enough!\n");
    }

    return 0;
}
```

Bài tập 10 (bonus): Tính tích hai ma trận vuông

Dưới đây cung cấp đoạn code đơn giản để tính tích của hai ma trận cỡ NxNNxN theo công thức trực tiếp.
Hãy viết hàm tính tích hai ma trận nhưng có tốc độ nhanh hơn ít nhất 10% so với code đơn giản.
Gợi ý: hãy để ý đến thứ tự truy cập các phần tử trong ma trận, tối ưu cache hoặc sử dụng thuật toán tốt hơn O(N3)O(N3).

In [ ]:

```
#include <iostream>
#include <cstring>

using namespace std;

const int N = 128;

struct Matrix {
    unsigned int mat[N][N];

    Matrix() {
        memset(mat, 0, sizeof mat);
    }
};

bool operator == (const Matrix &a, const Matrix &b) {
    for (int i = 0; i < N; ++i) {
        for (int j = 0; j < N; ++j) {
            if (a.mat[i][j] != b.mat[i][j]) return false;
        }
    }
    return true;
}

Matrix multiply_naive(const Matrix &a, const Matrix &b) {
    Matrix c;
    for (int i = 0; i < N; ++i) {
```

```
        for (int j = 0; j < N; ++j) {
            for (int k = 0; k < N; ++k) {
                c.mat[i][j] += a.mat[i][k] * b.mat[k][j];
            }
        }
    }
    return c;
}

Matrix multiply_fast(const Matrix &a, const Matrix &b) {
    /*****************
    # YOUR CODE HERE #
    *****************/
}

Matrix gen_random_matrix() {
    Matrix a;
    for (int i = 0; i < N; ++i) {
        for (int j = 0; j < N; ++j) {
            a.mat[i][j] = rand();
        }
    }
    return a;
}

Matrix base;

double benchmark(Matrix (*multiply) (const Matrix&, const Matrix&), Matrix
&result) {
    const int NUM_TEST = 10;
    const int NUM_ITER = 64;

    Matrix a = base;
    result = a;

    double taken = 0;
    for (int t = 0; t < NUM_TEST; ++t) {
        clock_t start = clock();
        for (int i = 0; i < NUM_ITER; ++i) {
            a = multiply(a, result);
            result = multiply(result, a);
        }
        clock_t finish = clock();
        taken += (double)(finish - start);
    }
    taken /= NUM_TEST;

    printf("Time: %.9f\n", taken / CLOCKS_PER_SEC);
    return taken;
}
```

```
int main() {
    base = gen_random_matrix();

    Matrix a, b;
    printf("Slow version\n");
    double slow = benchmark(multiply_naive, a);
    printf("Fast version\n");
    double fast = benchmark(multiply_fast, b);

    if (a == b) {
        printf("Correct answer! Your code is %.2f%% faster\n", slow / fast
* 100.0);
    } else {
        printf("Wrong answer!\n");
    }
    return 0;
}
```

Phần 3. Bài tập về nhà

Bài tập 11: Tính tích hai đa thức

Cho 2 đa thức A(x) và B(x) tương ứng có bậc N và M. Hãy tính ma trận tích C(x) = A(x) * B(x) có bậc $N+M-1$.

Input: Gồm 2 dòng biểu diễn các đa thức A(x) và B(x), mỗi dòng

- Số đầu tiên N là bậc của đa thức;
- $N+1$ số nguyên tiếp theo, số thứ i là hệ số của x^{i-1}.

Output: Một số nguyên duy nhất là XOR của các hệ số của đa thức C(x).

Ví dụ:

Input:
3 83 86 77 15
4 93 35 86 92 49

Output:
20731

Giải thích: các hệ số của đa thức kết quả lần lượt là 7719, 10903, 17309, 19122, 19126, 12588, 5153, 735.

Giới hạn:

- Các hệ số của các đa thức đầu vào có trị tuyệt đối nhỏ hơn 100.
- Có 5 tests, test thứ i có bậc của các đa thức đầu vào không quá 10^i.

Bài tập 12: Map Sort

Hôm nay, cô giáo giao cho An một câu hỏi hóc búa. Cô cho một danh sách với mỗi phần tử có dạng <key, value> và yêu cầu An sắp xếp danh sách đó giảm dần theo giá trị value. Nếu 2 phần tử có value giống nhau thì sắp xếp giảm dần theo key.

Hãy viết một chương trình sử dụng hàm nặc danh để giúp An làm bài tập.
Input: Danh sách đầu vào. Mỗi dòng ghi một cặp giá trị key, value cách nhau bởi dấu cách (|key| ≤10^9≤109, |value| ≤10^9≤109).

Output: In danh sách đã được sắp xếp theo yêu cầu. Mỗi dòng ghi một cặp giá trị key, value cách nhau bởi dấu cách.

Ví dụ:

Input:
2 3
4 8
9 1
1 8

Output:
4 8
1 8
2 3
9 1

Bài tập 13: Big Integer

Số nguyên lớn là các số nguyên có giá trị rất lớn và không thể biểu diễn bằng các kiểu dữ liệu nguyên cơ bản. Để biểu diễn số nguyên lớn, ta có thể dùng kiểu struct như sau:

```
struct bigNum{
    char sign;
    char num[101];
};
```

Nhiệm vụ các bạn là đa năng hóa các toán tử để thực hiện các phép toán số học với kiểu dữ liệu số nguyên lớn vừa định nghĩa ở trên.
Input: Dữ liệu vào gồm hai dòng mô tả hai số nguyên lớn aa và bb, mỗi dòng chứa 1 chuỗi ký tự mô tả 1 số nguyên lớn không vượt quá 10^{100}10100. Chữ số đầu của mỗi chuỗi ký tự sẽ thể hiện dấu của số đó: 0 là âm, 1 là dương. Các chữ số sau thể hiện giá trị của số đó.
Output: In ra giá trị của biểu thức $ab-3a+4b$ab−3a+4b. Kết quả in ra một số nguyên lớn dưới dạng chuỗi ký tự có định dạng như mô tả trong dữ liệu vào.

Ví dụ:

Input:
0121807015
1347227347

Output:
042294724910108772