Data Organization Topic: Exercise 1

- Assignee: Viet
- Status: Done
- Due: January 11, 2024
- Project: HCMUS (https://www.notion.so/HCMUS-dca25e32ad334f19ba250f3fc14e4ec1?pvs=21)
- Priority: High
- Spent time (Hours): 4.5

Thông tin sinh viên

- Họ và tên: Cao Hoài Việt
- MSSV: 22850034
- Email SV: 228050034@student.hcmus.edu.vn
- Email cá nhân: viet.ch2612@gmail.com

1. (2 đ). Bài tập 3.3.2.

Viết hàm tìm ước số chung lớn nhất của hai số nguyên dương. Sau đó dùng hàm này để tìm ước số chung lớn nhất trong n số nguyên dương.

```
#include <iostream>

int getHcf(int a, int b) {
    while (b != 0) {
        int tmp = b;
        b = a % b;
        a = tmp;
    }
    return a;
}

int getHcfInArray(int *arr) {
    int size = sizeof(&arr) / sizeof(arr[0]);
    int result = 0;
    for (int i = 0; i <= size; i++) {
        result = getHcf(result, arr[i]);
    }

    return result;
}

void printArray(int *arr) {
    int size = sizeof(&arr) / sizeof(arr[0]);
    for (int i = 0; i <= size; i++) {
        std::cout << arr[i] << ", ";
    }
}
```

Kết quả chạy thử

```
int main() {
    int arr[] = {12, 18, 24}; // 6
    printArray(arr);
    std::cout << " HCF: " << getHcfInArray(arr) << std::endl;

    int arr2[] = {36, 48, 60}; // 12
    printArray(arr2);
    std::cout << " HCF: " << getHcfInArray(arr2) << std::endl;

    int arr3[] = {15, 25, 35}; // 5
    printArray(arr3);
    std::cout << " HCF: " << getHcfInArray(arr3) << std::endl;

    return 0;
}

ex1 ❯❯❯ cd "/Users/vietcao/Documents/hcm-
assignment/data_organization/ex1/" && g++ -std=c++17 question_1.cpp -o
question_1 && "/Users/vietcao/Documents/hcm-
assignment/data_organization/ex1/"question_1
12, 18, 24,  HCF: 6
36, 48, 60,  HCF: 12
15, 25, 35,  HCF: 5
```

2. (2 đ). Bài tập 3.3.3.

Viết hàm tính số tổ hợp nChooseK bằng cách “tách bạch, ngây thơ” và cách “tích hợp, tinh vi” (xem Bài 3.2):

- Viết riêng hàm tính giai thừa và dùng hàm này khi tính
- Phân tích, biến đổi, … để tính So sánh ưu khuyến điểm của hai cách làm.

******Cách 1. Sử dụng cách “Tách bạch, ngây thơ”: Viết riêng hàm tính giai thừa (Đệ quy) và dùng hàm này tính tổ hợp.******

```
#include <iostream>

int factorial(int n) {
    if (n == 0 || n == 1) {
        return 1;
    }
    return n * factorial(n - 1);
}

int nChooseK(int n, int k) {
    if (n < k) {
        return 0;
    }
```

```
    return factorial(n) / (factorial(k) * factorial(n - k));
}
```

****Cách 2. Sử dụng cách "Tích hợp, tinh vi": Chúng ta sẽ sử dụng "Công thức tổ hợp" để tính.****

$$ C(n, k) = (n-k+1) * (n-k+2) * ... * (n-1) * n / (1 * 2 * ... * k)

$$

```
int nChooseKSmart(int n, int k) {
    if (n < k) {
        return 0;
    }

    int result = 1;
    for (int i = 1; i <= k; ++i) {
        result *= n - k + i;
        result /= i;
    }

    return result;
}
```

So sánh ưu, khuyết điểm của 2 cách.

Cách 1

Ưu điểm	Nhược điểm
- Dễ đọc, dễ hiểu.	- Hiệu suất thấp do phải tính giai thừa nhiều lần (3 lần).
- Code phản ánh trực tiếp công thức mà chúng ta được học.	- Có thể gặp lỗi tràn số với n, k lớn

Cách 2

Ưu điểm	Nhược điểm
- Hiệu suất cao hơn 3 lần so với cách 1 do giảm thiểu số lần tính toán.	- Khó hiểu khi không quen với các công thức toán học.
- Giảm rủi ro bị tràn số khi n, k lớn.	

Thử tính toán thời gian thực thi của 2 cách với $n=100000, k= 90000$

```
int main() {
    int n = 100000, r = 90000;

    auto startNaive = std::chrono::high_resolution_clock::now();
```

```
    nChooseK(n, r);
    auto stopNaive = std::chrono::high_resolution_clock::now();
    auto durationNaive =
std::chrono::duration_cast<std::chrono::microseconds>(
        stopNaive - startNaive);

    auto startSmart = std::chrono::high_resolution_clock::now();
    nChooseKSmart(n, r);
    auto stopSmart = std::chrono::high_resolution_clock::now();
    auto durationSmart =
std::chrono::duration_cast<std::chrono::microseconds>(
        stopSmart - startSmart);

    std::cout << std::endl;
    std::cout << "nChooseK: " << durationNaive.count() << "
microseconds\n";
    std::cout << "nChooseKSmart: " << durationSmart.count()
              << " microseconds\n";

    return 0;
}
```

Kết quả: Cách 2 có thời gian thực thi nhanh hơn 3 lần so với cách 1.

```
ex1 ❯❯❯ cd "/Users/vietcao/Documents/hcm-
assignment/data_organization/ex1/" && g++ -std=c++17 question_2.cpp -o
question_2 && "/Users/vietcao/Documents/hcm-
assignment/data_organization/ex1/"question_2

nChooseK: 3927 microseconds
nChooseKSmart: 1260 microseconds
```

3. (2 đ). Bài tập 3.5.5.

1. **Giá trị của `a, b, c, d` là bao nhiêu?** `a = 97, b = 2, c = 12.3, d = 2.5`
2. **Chi tiết các bước chuyển kiểu trong các dòng mã trên**

```
int a = 'a'; // a = 97
// Chuyển giá trị ký tự 'a' sang kiểu số nguyên int.
// Trong ASCII, ‘a’ có giá trị là 97 nên a = 97.

int b = (1 < a) + (a < 1000); // b = 2
// Đây là biểu thức logic nhưng kết quả được chuyển thành kiểu int.
// Với biểu thức logic, true == 1 và false == 0
// Nên kết quả sẽ là (1 < 97) + ( 97 < 1000) == true + true == 1 + 1 == 2

double c = 10.3 + 2;
// Chuyển giá trị 2 sang kiểu double và cộng với 10.3
// Kết quả là 12.3
```

```
float d = (float) 10 / 4;
// Chuyển giá trị 10 sang kiểu float và chia cho 4.
// Kết quả là 2.5 nhưng vì kiểu dữ liệu của d là float nên kết quả là 2.5f
```

c. Chuyển kiểu tường minh xảy ra ở những chỗ nào trong đoạn code trên?

Xảy ra ở dòng `float d = (float)10 / 4;`, giá trị `10` được chuyển từ kiểu số nguyên `int` sang kiểu `float` để thực hiện phép chia.

Nếu không ép kiểu/chuyển kiểu tường minh từ `int 10` sang `float 10` thì phép toán `10/4` sẽ cho ra kết quả `= 2` do kiểu `int` chỉ lấy phần nguyên.

4. (2 đ). Bài tập 1.5.10. So sánh kết quả tính bằng hàm trong thư viện chuẩn `cmath` của C/C++.

Dùng phương pháp chia đôi trong phần mở rộng để tính căn bậc 3 của một số thực dương.

```
#include <cmath>
#include <iostream>

double getAbsoluteValue(double x) {
    if (x < 0)
        return -x;
    return x;
}

double myCbrt(double y) {
    double mid, l, r;
    const double EPS = 0.0000001;

    if (y > 1)
        r = y;
    else
        r = 1;
    l = 0;

    do {
        mid = (l + r) / 2;
        if (mid * mid * mid > y)
            r = mid;
        else
            l = mid;
    } while (getAbsoluteValue(mid * mid * mid - y) > EPS);

    return mid;
}

int main() {
    double x = 3;

    std::cout << std::endl;
```

```
    std::cout << "mySqrt(" << x << ") = " << myCbrt(x) << std::endl;
    std::cout << "std::sqrt(" << x << ") = " << std::cbrt(x) << std::endl;

    return 0;
}
```

5. (2 đ). Viết hàm để tính giá trị $\sin(x)$ với x tính theo radian (không được dùng thư viện). So sánh kết quả với hàm sin trong thư viện chuẩn cmath của C/C++.

```
#include <cmath>
#include <iostream>

double factorial(int n) {
    if (n == 0 || n == 1) {
        return 1;
    }
    return n * factorial(n - 1);
}

double sin(double x) {
    double result = 0;
    for (int i = 0; i < 10; ++i) {
        result += pow(-1, i) * pow(x, 2 * i + 1) / factorial(2 * i + 1);
    }
    return result;
}

int main() {
    double x = 0.5;

    std::cout << std::endl;
    std::cout << "sin(" << x << ") = " << sin(x) << std::endl;
    std::cout << "std::sin(" << x << ") = " << std::sin(x) << std::endl;

    return 0;
}
```

Chạy và kiểm tra kết quả. Giống nhau.

```
ex1 ❯❯❯ cd "/Users/vietcao/Documents/hcm-
assignment/data_organization/ex1/" && g++ -std=c++17 question_5.cpp -o
question_5 && "/Users/vietcao/Documents/hcm-
assignment/data_organization/ex1/"question_5

sin(0.5) = 0.479426
std::sin(0.5) = 0.479426
```