Trạng thái	Đã xong
Bắt đầu vào lúc	Thứ Tư, 16 tháng 4 2025, 4:27 PM
Kết thúc lúc	Thứ Tư, 16 tháng 4 2025, 4:50 PM
Thời gian thực hiện	23 phút 26 giây

Câu hỏi 1

Đúng

Đạt điểm 1,00

[Tiếng Việt]

Viết hàm tìm chữ cái xuất hiện nhiều lần nhất trong một chuỗi và số lần xuất hiện của chữ cái đó. Nếu có nhiều chữ cái thỏa mãn, xác định chữ cái có thứ tự alphabet nhỏ nhất.

Đầu vào:

- const char* str: chuỗi ký tự đang xét, chỉ bao gồm chữ cái thường và hoa

Đầu ra:

- char& res: chữ cái xuất hiện nhiều lần nhất trong chuỗi str
- int& freq: số lần xuất hiện của chữ cái đó

Chú ý: tham số res là chữ cái thường. Chữ cái viết hoa strong str cũng được tính là giống chữ cái viết thường (VD "aA" chứa hai ký tự 'a')

Lưu ý: Sinh viên chỉ có 5 lần nộp không tính penalty, ở lần nộp thứ 6 trở đi bài làm sẽ được tính là 0 điểm.

[English]

Write a function that finds the most frequent character in a given string and how many times that character appears. If there are multiple characters satisfy the condition, find the one with the lowest alphabetical order.

Input:

- const char* str: the input string, contains only lowercase and uppercase ASCII letters

Output:

- char& res: the most frequent character in str
- int& freq: the times that the result character appears in the string

Note: return res as a lowercase letter. Uppercase letters in str is also counted as lowercase letters (e.g the string "aA" contains two letter 'a')

For example:

Test	Input	Result

int n;
cin >> n;
char* str = new char[n+1];
for(int i = 0; i < n; i++) cin >> str[i];
str[n] = 0;
char res = 0;
int freq = 0;
mostFrequentCharacter(str, res, freq);
cout << res << ' ' << freq;
delete[] str;
	7
	aCxAboc
		a 2

Test	Input	Result

int n;
cin >> n;
char* str = new char[n+1];
for(int i = 0; i < n; i++) cin >> str[i];
str[n] = 0;
char res = 0;
int freq = 0;
mostFrequentCharacter(str, res, freq);
cout << res << ' ' << freq;
delete[] str;
	6
	OmnPqB
		b 1

Answer: (penalty regime: 0, 0, 0, 0, 0, 0, 0, 0, 50, 100 %)

Reset answer

```
/**
 * Hàm tìm chữ cái xuất hiện nhiều nhất trong chuỗi và số lần xuất hiện của nó
 * @param str: Con trỏ tới chuỗi cần kiểm tra (chỉ chứa chữ cái ASCII)
 * @param res: Biến tham chiếu để lưu kết quả chữ cái xuất hiện nhiều nhất (ở dạng chữ thường)
 * @param freq: Biến tham chiếu để lưu số lần xuất hiện của chữ cái kết quả
 */
void mostFrequentCharacter(const char* str, char& res, int& freq) {
    // Khởi tạo mảng đếm cho 26 chữ cái từ 'a' đến 'z'
    // count[0] tương ứng với 'a', count[1] tương ứng với 'b', ...
    int count[26] = {0};

    // Duyệt qua từng ký tự trong chuỗi đầu vào
    for (int i = 0; str[i] != '\0'; i++) {
        char c = str[i];

        // Chuyển đổi chữ cái hoa thành chữ cái thường để đếm
        if (c >= 'A' && c <= 'Z') {
            c = c - 'A' + 'a';  // Công thức chuyển đổi: c = c - 65 + 97
        }

        // Chỉ xét các ký tự là chữ cái
        if (c >= 'a' && c <= 'z') {
            // Tăng số đếm cho chữ cái này
            // c - 'a' cho vị trí index trong mảng count
            count[c - 'a']++;
        }
    }

    // Khởi tạo biến để tìm chữ cái xuất hiện nhiều nhất
    int maxCount = 0;        // Số lần xuất hiện nhiều nhất
    char maxChar = 'a';      // Chữ cái xuất hiện nhiều nhất

    // Duyệt qua mảng đếm để tìm chữ cái xuất hiện nhiều nhất
    for (int i = 0; i < 26; i++) {
        // Nếu số lần xuất hiện lớn hơn maxCount hiện tại
        // Hoặc bằng nhau nhưng đã có sẵn chữ cái thứ tự nhỏ nhất do ta duyệt từ 'a' đến 'z'
        if (count[i] > maxCount) {
            maxCount = count[i];         // Cập nhật số lần xuất hiện lớn nhất
            maxChar = 'a' + i;           // Cập nhật chữ cái tương ứng
        }
        // Không cần xét trường hợp count[i] == maxCount vì ta đã duyệt theo thứ tự alphabet
        // nên mặc định nếu bằng nhau thì chữ cái có thứ tự alphabet nhỏ hơn đã được chọn trước đó
    }

    // Gán kết quả vào các biến tham chiếu
    res = maxChar;   // Chữ cái xuất hiện nhiều nhất (luôn là chữ thường)
    freq = maxCount; // Số lần xuất hiện của chữ cái đó
}
```

	Test	Input	Expected	Got	

✓	int n;
	cin >> n;
	char* str = new char[n+1];
	for(int i = 0; i < n; i++) cin >> str[i];
	str[n] = 0;
	char res = 0;
	int freq = 0;
	mostFrequentCharacter(str, res, freq);
	cout << res << ' ' << freq;
	delete[] str;
		7
		aCxAboc
			a 2	a 2	✓

✓	int n;
	cin >> n;
	char* str = new char[n+1];
	for(int i = 0; i < n; i++) cin >> str[i];
	str[n] = 0;
	char res = 0;
	int freq = 0;
	mostFrequentCharacter(str, res, freq);
	cout << res << ' ' << freq;
	delete[] str;
		6
		OmnPqB
			b 1	b 1	✓

Passed all tests! ✓

Câu hỏi **2**

Đúng

Đạt điểm 1,00

[Tiếng Việt]

Cho hai số tự nhiên n và m. Hãy viết một hàm tính toán và trả về giá trị của n trong hệ m phân, biết giá trị m thuộc {2, 4, 8}. Sinh viên cần thay đổi tham số m để có giá trị mặc định là 2 trong định nghĩa hàm.

Đầu vào:

int n: số tự nhiên n cần được chuyển đổi sang hệ m phân

int m: hệ cơ số cần chuyển đổi. m phải có đối số mặc định là 2 khi định nghĩa hàm.

Đầu ra:

int: giá trị của n trong hệ m phân

Chú ý: đầu vào sẽ đảm bảo giá trị trả về không bị tràn số

[English]

Given two natural numbers n and m. Write a function that converts n to base m and returns that value, given that m is in {2, 4, 8}. Student must provide a default argument of value 2 for the second parameter(m).

Input:

int n: a natural number n that needs to be converted

int m: the new base which n is converted to. The default argument of this parameter must be 2 when defining the function.

Output:

int: the value of n in base m

Note: the input guarantees that the return value will not cause integer overflow

For example:

Test	Input	Result

`int n;`
`cin >> n;`
`cout << convertToBaseM(n, 4) << '\n';`
`cout << convertToBaseM(n, 8) << '\n';`
`cout << convertToBaseM(n);`
	16	100
		20
		10000

`int n;`
`cin >> n;`
`cout << convertToBaseM(n, 4) << '\n';`
`cout << convertToBaseM(n, 8) << '\n';`
`cout << convertToBaseM(n);`
	28	130
		34
		11100

Answer: (penalty regime: 0, 0, 0, 0, 0, 100 %)

Reset answer

```
/**
 * Hàm chuyển đổi số nguyên n từ hệ thập phân sang hệ cơ số m
 * @param n: Số nguyên cần chuyển đổi
 * @param m: Cơ số cần chuyển đổi sang (m chỉ có thể là 2, 4 hoặc 8), mặc định là 2
 * @return Giá trị của n trong hệ cơ số m
 */
int convertToBaseM(int n, int m = 2) {
    // Kiểm tra trường hợp cơ sở: nếu n = 0 thì kết quả là 0 trong bất kỳ hệ cơ số nào
    if (n == 0) {
        return 0;
    }

    // Khai báo biến để lưu kết quả và vị trí số mũ
    int result = 0;     // Biến lưu kết quả cuối cùng
    int position = 1;   // Biến đại diện cho vị trí (hàng đơn vị, hàng chục, ...)

    // Duyệt qua từng chữ số của n và chuyển đổi sang hệ cơ số m
    while (n > 0) {
```

```
        // Lấy phần dư khi chia n cho m (tương đương với chữ số ở vị trí hiện tại)
        int remainder = n % m;

        // Thêm chữ số vào kết quả (nhân với vị trí thích hợp)
        result += remainder * position;

        // Di chuyển sang vị trí tiếp theo (nhân position với 10)
        position *= 10;

        // Chia n cho m để xét chữ số tiếp theo
        n /= m;
    }

    return result;
}
```

	Test	Input	Expected	Got	
✓	int n; cin >> n; cout << convertToBaseM(n, 4) << '\n'; cout << convertToBaseM(n, 8) << '\n'; cout << convertToBaseM(n);	16	100 20 10000	100 20 10000	✓
✓	int n; cin >> n; cout << convertToBaseM(n, 4) << '\n'; cout << convertToBaseM(n, 8) << '\n'; cout << convertToBaseM(n);	28	130 34 11100	130 34 11100	✓

Passed all tests! ✓