



Data Org Ex2

| | |
|----------------------|--------------------|
| 👤 Assignee | 🧑 Viet |
| ⚙️ Status | Done |
| 📅 Due | @February 24, 2024 |
| 🏠 Project | 🏠 HCMUS |
| 📌 Priority | Medium |
| # Spent time (Hours) | 3.5 |

Thông tin sinh viên

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

Câu 1. Viết lại chương trình Bầu Cua sử dụng Array

```
#include <iostream>
```

```

using namespace std;

#define DUNG 0
#define RUNG 7
#define TOTAL 6

string ten[TOTAL] = {"Tom", "Cua", "Bau", "Ca", "Ga", "Nai"};

void xuat(int cuoc[]) {
    int tongCuoc = 0;

    for (int i = 0; i < TOTAL; i++) tongCuoc += cuoc[i];
    if (tongCuoc == 0) {
        cout << "Ban chua dat cuoc!" << endl;
    } else {
        cout << "Ban dang dat: ";
        for (int i = 0; i < TOTAL; i++)
            if (cuoc[i] > 0) cout << ten[i] << ": " << cuoc[i] << "; ";
        cout << endl;
    }

    cout << "Ban muon (0-Dung, 1-Tom, 2-Cua, 3-Bau, 4-Ca, 5-Ga, 6-Nai): ";
}

void xuatTongCuoc(int cuoc[]) {
    int kq = 0;
    for (int i = 0; i < TOTAL; i++) kq += cuoc[i];
    cout << "Tong cuoc: " << kq << endl;
}

void rung(int kqRung[]) {
    cout << "KQ rung: ";
    for (int i = 0; i < 3; i++) {
        int t = rand() % 6;
        kqRung[i] = t;
        cout << ten[t] << " ";
    }
}

```

```

    }
    cout << endl;
}

void xuatKqRung(int cuoc[], int kqRung[]) {
    int result = 0;
    for (int i = 0; i < 3; i++) {
        result += cuoc[kqRung[i]];
    }
    for (int i = 0; i < TOTAL; i++) cuoc[i] = 0;

    cout << "Ban con lai " << result << endl;
}

int main() {
    int cmd;
    int cuoc[TOTAL] = {0};
    int kqRung[3] = {0};

    srand(time(0));
    while (1) {
        xuat(cuoc);
        cin >> cmd;

        if (cmd == DUNG) {
            cout << "Chuc mung nam moi!" << endl;
            return 0;
        }

        if (cmd >= 1 && cmd <= 6) {
            cout << "Ban muon dat " << ten[cmd - 1] << " bao nhieu tien" << endl;
            cin >> cuoc[cmd - 1];
        }

        if (cmd == RUNG) {
            xuatTongCuoc(cuoc);
        }
    }
}

```

```

        rung(kqRung);
        xuấtKqRung(cuoc, kqRung);
        return 0;
    }
}

return 0;
}

```

Kết quả chạy

```

1 #include <iostream>
2 using namespace std;
3 #define DUNG 0
4 #define RUNG 7
5 #define TOTAL 6
6 string ten[TOTAL] = {"Tom", "Cua", "Bau", "Ca", "Ga", "Nai"};
7 void xuất(int cuoc[]) {
8     int tongCuoc = 0;
9     for (int i = 0; i < TOTAL; i++) tongCuoc += cuoc[i];
10    if (tongCuoc == 0) {
11        cout << "Ban chua dat cuoc!" << endl;
12    } else {
13        cout << "Ban dang dat: ";
14        for (int i = 0; i < TOTAL; i++)
15            if (cuoc[i] > 0) cout << ten[i] << ": " << cuoc[i] << " ";
16        cout << endl;
17    }
18    cout << "Ban muon (0-Dung, 1-Tom, 2-Cua, 3-Bau, 4-Ca, 5-Ga, 6-Nai, 7-
19    -Rung): ";
20 }
21 void xuấtTongCuoc(int cuoc[]) {
22     int kq = 0;
23     for (int i = 0; i < TOTAL; i++) kq += cuoc[i];
24     cout << "Tong cuoc: " << kq << endl;
25 }
26 void rung(int kqRung[]) {
27     cout << "KQ rung: ";
28     for (int i = 0; i < 3; i++) {
29         int t = rand() % 6;
30         kqRung[i] = t;
31         cout << ten[t] << " ";
32     }
33     cout << endl;
34 }
35 void xuấtKqRung(int cuoc[], int kqRung[]) {
36     int result = 0;
37     for (int i = 0; i < 3; i++) {
38         result += cuoc[kqRung[i]];
39     }
40     for (int i = 0; i < TOTAL; i++) cuoc[i] = 0;
41     cout << "Ban con lai " << result << endl;
42 }
43 int main() {

```

Câu 2. Tìm điểm yên ngựa

```

#include <iostream>

int main() {
    // Input 2D array
    int m, n;

```

```

std::cout << "So hang: ";
std::cin >> m;
std::cout << "So cot: ";
std::cin >> n;
int a[m][n];
std::cout << "Bat dau nhap cac phan tu cua mang." << std::endl;
for (int i = 0; i < m; i++) {
    std::cout << "a[" << i << "][?]: ";
    for (int j = 0; j < n; j++) {
        std::cin >> a[i][j];
    }
}

// Start finding saddle points
// Create variables to locations and number of points
int saddlePointsCount = 0;
int saddlePoints[m][n];
for (int i = 0; i < m; i++)
    for (int j = 0; j < n; j++) saddlePoints[i][j] = 0;

// Loop through each row
for (int i = 0; i < m; i++) {
    int min = a[i][0]; // init min = first element
    int minCol = 0;    // min column index

    // At row i, go through from left to right to find the min
    for (int j = 1; j < n; j++) {
        if (a[i][j] < min) {
            min = a[i][j];
            minCol = j;
        }
    }

    bool isSaddlePoint = true; // Assume it is true then validate
    // Loop through each element in column minCol from 0 to m

```

```

// If it is not the highest value, break
for (int k = 0; k < m; k++) {
    if (a[k][minCol] > min) {
        isSaddlePoint = false;
        break;
    }
}

// Save the result
if (isSaddlePoint) {
    saddlePoints[i][minCol] = 1; // let assume 1 means true
    saddlePointsCount++;
}
}

std::cout << "Hien thi lai ma tran 2D: " << std::endl;
for (int i = 0; i < m; i++) {
    for (int j = 0; j < n; j++) {
        std::cout << a[i][j] << " ";
    }
    std::cout << std::endl;
}
std::cout << std::endl;

// Output saddle points
std::cout << "Cac diem yen ngua cua mang: " << std::endl;
for (int i = 0; i < m; i++) {
    for (int j = 0; j < n; j++) {
        if (saddlePoints[i][j] == 1) {
            std::cout << "Diem yen ngua: a[" << i << ", " << j << "]"
                << std::endl;
        }
    }
}

std::cout << "So luong diem yen ngua: " << saddlePointsCount << std::endl;

```

```

return 0;
}

```

Kết quả chạy thử

```

1 #include <iostream>
2
3 int main() {
4     // Input 2D array
5     int m, n;
6     std::cout << "So hang: ";
7     std::cin >> m;
8     std::cout << "So cot: ";
9     std::cin >> n;
10    int a[m][n];
11    std::cout << "Bat dau nhap cac phan tu cua mang." << std::endl;
12    for (int i = 0; i < m; i++) {
13        std::cout << "a[" << i << "]?: ";
14        for (int j = 0; j < n; j++) {
15            std::cin >> a[i][j];
16        }
17    }
18
19    // Start finding saddle points
20    // Create variables to locations and number of points
21    int saddlePointsCount = 0;
22    int saddlePoints[m][n];
23    for (int i = 0; i < m; i++)
24        for (int j = 0; j < n; j++) saddlePoints[i][j] = 0;
25
26    // Loop through each row
27    for (int i = 0; i < m; i++) {
28        int min = a[i][0]; // init min = first element
29        int minCol = 0; // min column index
30
31        // At row i, go through from left to right to find the minimum col
32        for (int j = 1; j < n; j++) {
33            if (a[i][j] < min) {
34                min = a[i][j];
35                minCol = j;
36            }
37        }
38
39        bool isSaddlePoint = true; // assume it is true then validate it
40
41        // Loop through each element in column minCol from 0 to m
42        // If it is not the highest value, break
43        for (int k = 0; k < m; k++) {
44            if (a[k][minCol] > min) {
45                isSaddlePoint = false;
46                break;
47            }
48        }
49
50        // Save the result
51        if (isSaddlePoint) {
52            saddlePoints[i][minCol] = 1; // let assume 1 means true
53            saddlePointsCount++;
54        }
55    }
56
57    std::cout << "So diem yen ngua: " << saddlePointsCount << std::endl;
58
59    // Print the locations of saddle points
60    for (int i = 0; i < m; i++) {
61        for (int j = 0; j < n; j++) {
62            if (saddlePoints[i][j] == 1) {
63                std::cout << "Diem yen ngua: a[" << i << "][<< j << "] = " << a[i][j] << std::endl;
64            }
65        }
66    }
67
68    return 0;
69 }

```

Câu 3. Xóa các trùng nhau bằng cả 2 kĩ thuật: in-place và out-of-place. So sánh 2 kĩ thuật trong trường hợp này.

Kĩ thuật 1. **In-place.**

Kĩ thuật này em sẽ sử dụng biến index mới để tính lại index cho mảng, loại bỏ những vị trí bị trùng và sau đó sẽ xếp lại các vị trí trong mảng gốc và độ dài **n** dựa trên index này.

Ví dụ ta có mảng như sau:

[1, 2, 2, 3, 4], n = 5

Khi bắt đầu, **i** và **index** đều = 0, chỉ khi gặp value không phải duplicate thì index mới tăng.

Sau khi kết thúc vòng lặp, mảng mới sẽ có giá trị như sau

[1, 2, 3, 4, 4], n = 4. a[5] được coi là giá trị rác.

```
void remove_duplicate_values_in_place(int a[], int &n) {
    int index = 0; // init a new index

    for (int i = 0; i < n; i++) {
        bool isDuplicate = false;

        // Loop from 0 to index to see if a[k] is occurred or not
        for (int k = 0; k < index; k++) {
            if (a[i] == a[k]) {
                isDuplicate = true;
                break;
            }
        }

        // If a[i] is not duplicated, re-update its index
        if (!isDuplicate) a[index++] = a[i];
    }

    // Update the size of the array after we removed the duplicate
    n = index;
}
```

Kết quả chạy thử


```

main.cpp x remove_duplica.. x zsh x
ex1
ex2
  bau_cua.cpp
  remove_duplicates
  remove_duplicates.cpp
  roman_to_int.cpp
  saddle_points
  saddle_points.cpp
  BkipC_Q1.pdf
  main.cpp

28 n = index;
29 }
30
31 void remove_duplicate_values_out_place(int a[], int &n) {
32     int newArray[n];
33     int index = 0;
34
35     for (int i = 0; i < n; i++) {
36         bool isDuplicate = false;
37
38         for (int k = 0; k < index; k++) {
39             if (a[i] == newArray[k]) {
40                 isDuplicate = true;
41                 break;
42             }
43         }
44
45         if (!isDuplicate) {
46             newArray[index++] = a[i];
47         }
48     }
49
50     // Update the array
51     for (int i = 0; i < index; i++) {
52         a[i] = newArray[i];
53     }
54     // Update the size of the array
55     n = index;
56 }
57
58 int main() {
59     std::cout << "In-place" << std::endl;
60
61     int n;
62     std::cout << "Nhap do dai cua mang n: ";
63     std::cin >> n;
64
65     std::cout << "Vui long nhap gia tri cua mang a: " << std::endl;
66     int a[n];
67     for (int i = 0; i < n; i++) {
68         std::cin >> a[i];
69     }
70     std::cout << std::endl;
71
72     std::cout << "Original: " << std::endl;
73     print_array(a, n);
74
75     // Remove
76     std::cout << "Removed: " << std::endl;
77     remove_duplicate_values_in_place(a, n);
78     print_array(a, n);
79
80     return 0;
81 }

1 ex2 >>> g++ remove_duplicates.cpp -o remove_duplicates && ./remove_duplicates
2 In-place
3 Nhap do dai cua mang n: 14
4 Vui long nhap gia tri cua mang a:
5 1
6 3
7 58
8 2
9 5
10 1
11 3
12 5
13 5
14 58
15 009
16 8
17 204
18 37
19
20 Original:
21 1 3 58 2 5 1 3 5 5 58 9 8 204 37
22 Removed:
23 1 3 58 2 5 9 8 204 37
24 ex2 >>>
25
26

K TERMINAL zsh
~/Documents/hcm-assignment/data_organization/ex2/remove_duplicates.cpp" 81L, 1655B written
~/repos/E2E-WEB-PROD /~/repos/E2E-WEB-PROD Nvim ~ / vietcao@192:~
```

Kĩ thuật 2. Out-of-place

Với kĩ thuật này, em cũng sẽ sử dụng biến index tương tự và sẽ sử dụng thêm 1 mảng mới `newArray` rồi cập nhật `a` bằng `newArray`

```

void remove_duplicate_values_out_place(int a[], int &n) {
    int newArray[n];
    int index = 0;

    for (int i = 0; i < n; i++) {
        bool isDuplicate = false;

        for (int k = 0; k < index; k++) {
            if (a[i] == newArray[k]) {
                isDuplicate = true;
                break;
            }
        }

        // If a[i] is not duplicate, insert a[i] to the newArray
        // Then increase index by 1
        if (!isDuplicate) {
            newArray[index++] = a[i];
        }
    }

    // Then update a[] using newArray[]
    for (int i = 0; i < index; i++) {
        a[i] = newArray[i];
    }
    // Update the size n (virtual) of the a[].
    // The array's size (number of elements) can't be changed.
    n = index;
}

```

Kết quả chạy thử

```

28     n = index;
29 }
30
31 void remove_duplicate_values_out_place(int a[], int &n) {
32     int newArray[n];
33     int index = 0;
34
35     for (int i = 0; i < n; i++) {
36         bool isDuplicate = false;
37
38         for (int k = 0; k < index; k++) {
39             if (a[i] == newArray[k]) {
40                 isDuplicate = true;
41                 break;
42             }
43         }
44
45         if (!isDuplicate) {
46             newArray[index++] = a[i];
47         }
48     }
49
50     // Update the array
51     for (int i = 0; i < index; i++) {
52         a[i] = newArray[i];
53     }
54     // Update the size of the array
55     n = index;
56 }
57
58 int main() {
59     std::cout << "Out-place" << std::endl;
60
61     int n;
62     std::cout << "Nhap do dai cua mang n: ";
63     std::cin >> n;
64
65     std::cout << "Vui long nhap gia tri cua mang a: " << std::endl;
66     int a[n];
67     for (int i = 0; i < n; i++) {
68         std::cin >> a[i];
69     }
70     std::cout << std::endl;
71
72     std::cout << "Original: " << std::endl;
73     print_array(a, n);
74
75     // Remove
76     std::cout << "Removed: " << std::endl;
77     remove_duplicate_values_out_place(a, n);
78     print_array(a, n);
79
80     return 0;
81 }

```

```

1 ex2 >>> g++ remove_duplicates.cpp -o remove_duplicates && ./remove_duplicates
2 Out-place
3 Nhap do dai cua mang n: 10
4 Vui long nhap gia tri cua mang a:
5 1
6 2
7 4
8 2
9 6
10 8
11 10
12 23
13 3
14
15
16 Original:
17 1 2 4 2 6 8 10 23 3
18 Removed:
19 1 2 4 6 8 10 23 3
20 ex2 >>>

```

So sánh 2 cách chạy

| Kĩ thuật | Ưu điểm | Nhược điểm |
|----------|--|---|
| In-place | - Tiết kiệm bộ nhớ hơn do không cần thêm một mảng mới. | - Phức tạp hơn. - Dữ liệu gốc của bảng bị thay đổi |

| Kĩ thuật | Ưu điểm | Nhược điểm |
|-----------|---|---|
| | - Hiệu suất cao hơn: Vì không phải lưu bảng mới nên thời gian thực thi nhanh hơn. | trong quá trình thực thi. Nếu xảy ra lỗi trong quá trình thực thi thì dễ gây ra khả năng sai/mất dữ liệu. |
| Out-place | - Logic đơn giản hơn - Mảng gốc được giữ nguyên vẹn trước khi tìm ra kết quả trọn vẹn. | - Cần thêm bộ nhớ bổ sung - Hiệu suất thấp hơn |

Câu 4. Tra cứu để biết về chuỗi số La Mã (Roman numeral) và viết hàm

roman để tính giá trị của một chuỗi số La Mã. Ví dụ:
roman("MDCLXVI") trả về số nguyên 1666.

Cài đặt bằng C++

```
// Function to return the value of an individual Roman char
int value_of_roman(char c) {
    switch (c) {
        case 'I':
            return 1;
        case 'V':
            return 5;
        case 'X':
            return 10;
        case 'L':
            return 50;
        case 'C':
            return 100;
        case 'D':
            return 500;
        case 'M':
            return 1000;
        default:
            return -1;
    }
}
```

```

}

int roman_to_int(std::string s) {
    int result = 0;

    // If s[i] >= s[i+1] then result += s[i]
    // If s[i] < s[i+1] then result += s[i] - s[i+1]
    // And ignore the value at a[i+1]
    for (int i = 0; i < s.length(); i++) {
        int num1 = value_of_roman(s[i]);

        if (i < s.length() - 1) { // If it's not the last char
            int num2 = value_of_roman(s[i + 1]);

            if (num1 >= num2) {
                result += num1;
            } else {
                result += num2 - num1;
                i++; // Ignore the next i
            }
        } else { // If it's the last char
            result += num1;
        }
    }

    return result;
}

```

Kết quả chạy

```

1 #include <iostream>
2
3 int value_of_roman(char c) {
4     switch (c) {
5         case 'I':
6             return 1;
7         case 'V':
8             return 5;
9         case 'X':
10            return 10;
11         case 'L':
12            return 50;
13         case 'C':
14            return 100;
15         case 'D':
16            return 500;
17         case 'M':
18            return 1000;
19         default:
20            return -1;
21     }
22 }
23
24 int roman_to_int(std::string s) {
25     int result = 0;
26
27     // If s[i] >= s[i+1] then result += s[i]
28     // If s[i] < s[i+1] then result += s[i] - s[i+1]
29     // And ignore the value at s[i+1]
30     for (int i = 0; i < s.length(); i++) {
31         int num1 = value_of_roman(s[i]);
32
33         if (i < s.length() - 1) { // Don't need to check if it's the last
34             char
35                 int num2 = value_of_roman(s[i + 1]);
36             if (num1 >= num2) {
37                 result += num1;
38             } else {
39                 result += num2 - num1;
40                 i++; // Ignore the next i
41             }
42             } else {
43                 result += num1;
44             }
45         }
46     }
47     return result;
48 }
49
50 int main() {
51     // Exercise 4.4.7 / Question 4.
52     std::cout << "Vui long nhap so Roman: ";
53     std::string roman;
54
55     ex2 >> g++ roman_to_int.cpp -o roman_to_int 66 ./roman_to_int
56     Vui long nhap so Roman: MCMXCIV
57     So roman MCMXCIV tuong ung: 1994
58     ex2 >> ./roman_to_int
59     Vui long nhap so Roman: MCMXC
60     So roman MCMXC tuong ung: 1990
61     ex2 >> ./roman_to_int
62     Vui long nhap so Roman: LVIII
63     So roman LVIII tuong ung: 58
64     ex2 >> ./roman_to_int
65     Vui long nhap so Roman: LXVI
66     So roman LXVI tuong ung: 66
67     ex2 >> ./roman_to_int
68     Vui long nhap so Roman: XXXIX
69     So roman XXXIX tuong ung: 39
70     ex2 >> rm roman_to_int
71     ex2 >>

```