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

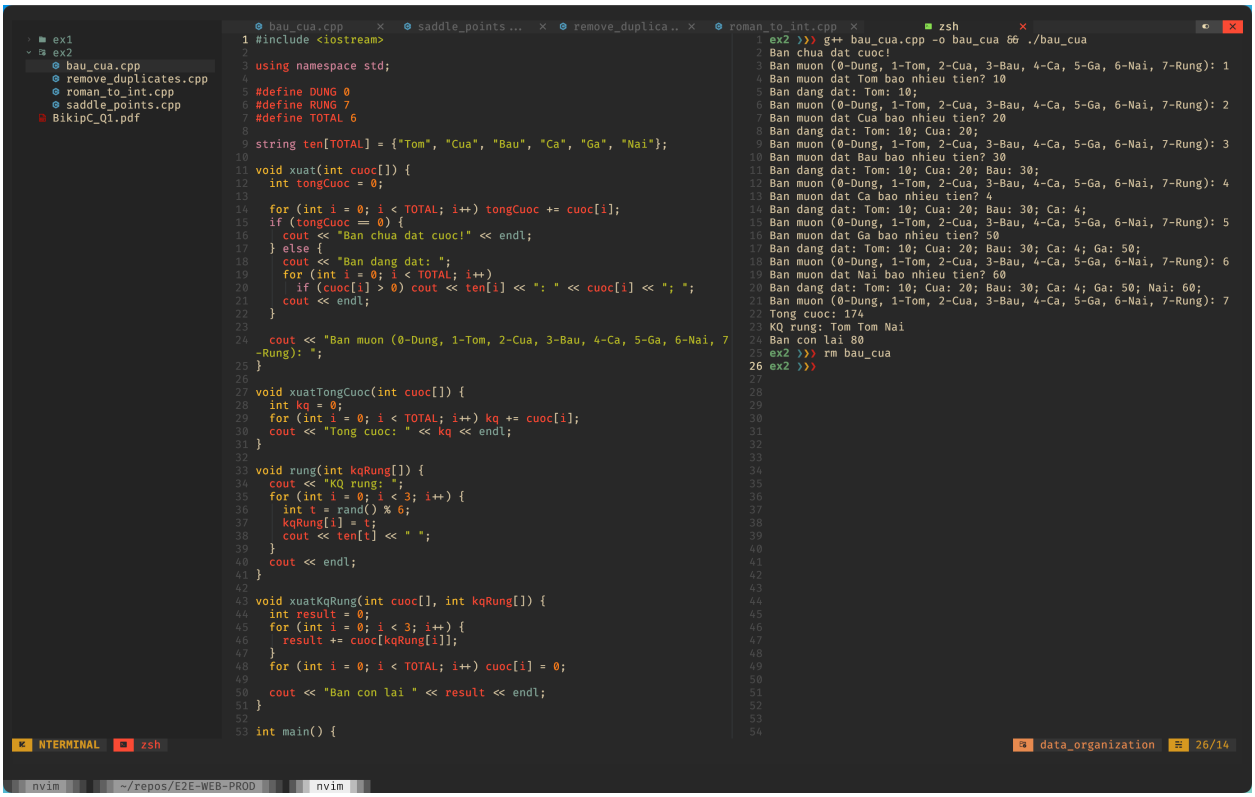
```
#include <iostream>
```

```
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-
}

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 tie
      cin >> cuoc[cmd - 1];
    }

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

```
        rung(kqRung);
        xuatKqRung(cuoc, kqRung);
        return 0;
    }
  }

  return 0;
}
```

Kết quả chạy



Câu 2. Tìm điểm yên ngựa

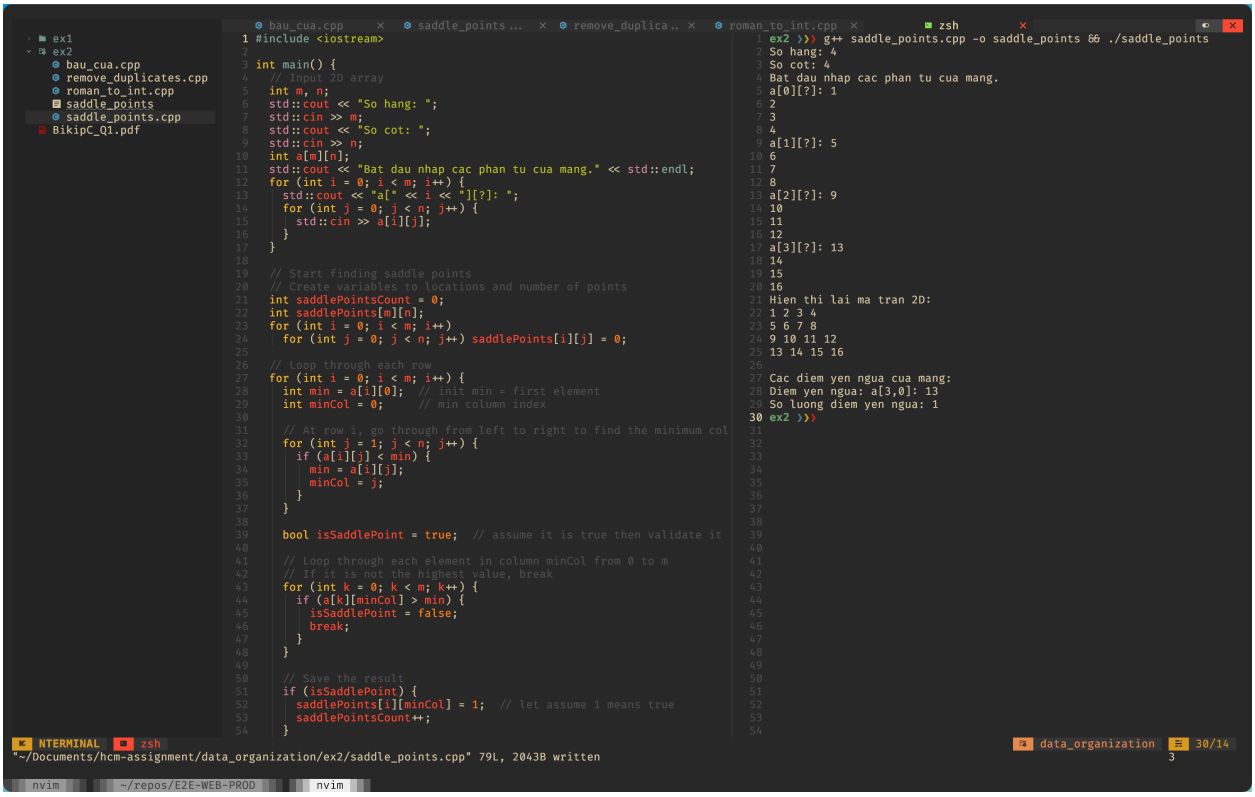
```
#include <iostream>

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 minir
      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 valida

      // 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
```

```
    return 0;
}
```

Kết quả chạy thử



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`

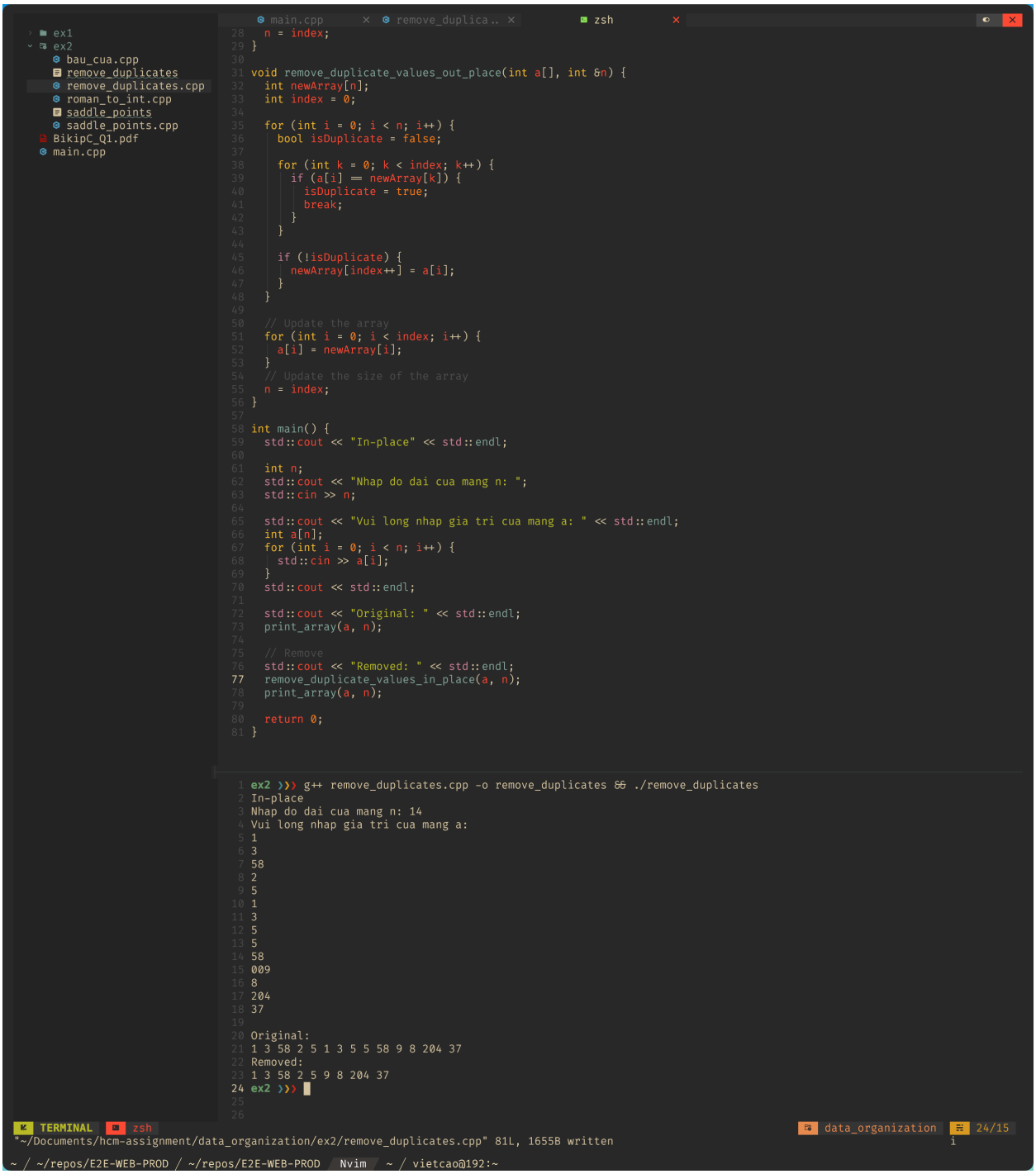
Khi bắt đầu, `i` và `index` đều = 0, chỉ khi gặp value không phải duplicate thì index mới tăng.

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 if a[k] is occured 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

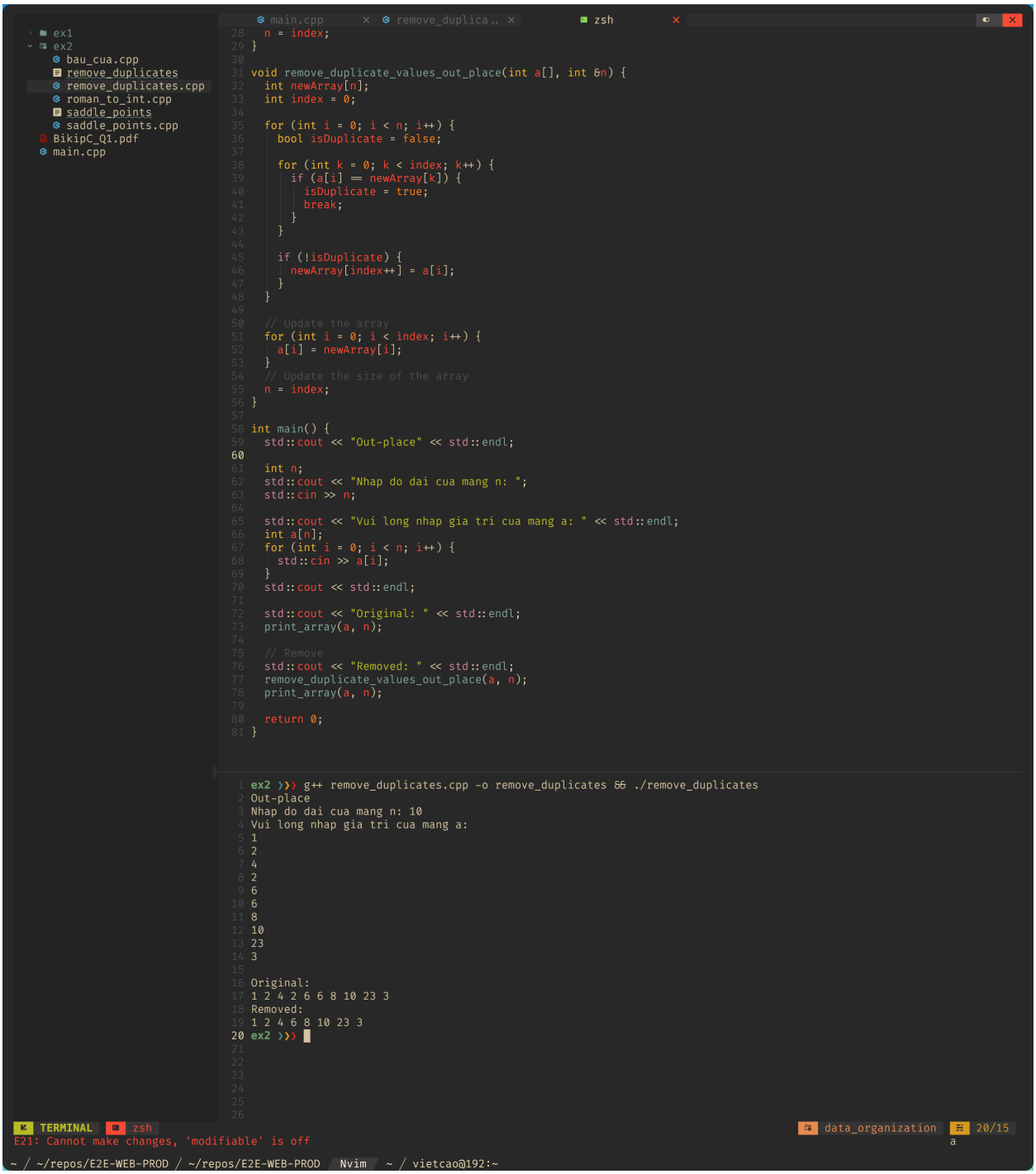


Kĩ thuật 2. `Out-of-place`

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



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

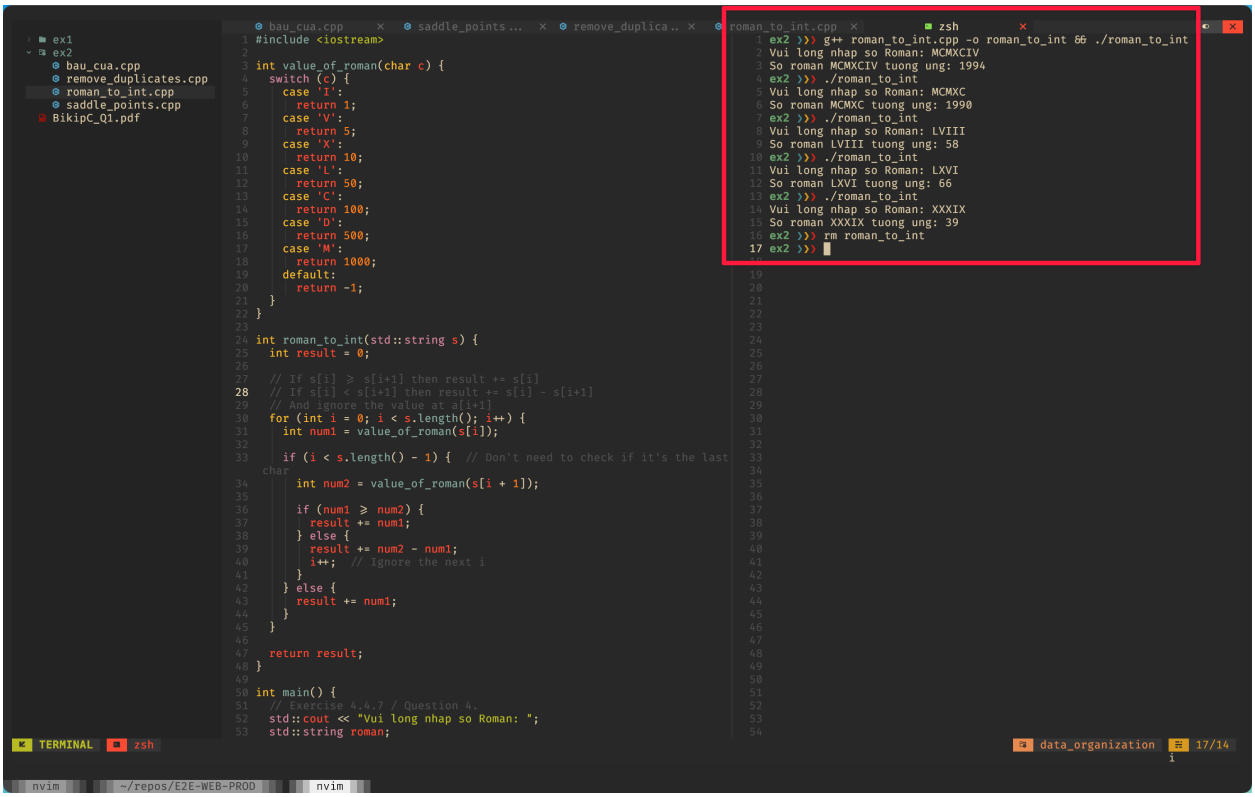
Câu 4. Tra cứu để biết về chuỗi số La Mã (Roman numeral) và viết hàm roman để tính giá trị của một chuỗi số La Mã. Ví dụ: roman("MDCLXVI") trả về số nguyên 1666.

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


ex1
ex2
bau_cua.cpp
remove_duplicates.cpp
roman_to_int.cpp
saddle_points.cpp
BikipC_Q1.pdf
#include <iostream>
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 s[i+1]
for (int i = 0; i < s.length(); i++) {
int num1 = value_of_roman(s[i]);
if (i < s.length() - 1) { // Don't need to check if it's 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 {
result += num1;
}
}
return result;
}
int main() {
// Exercise 4.4.7 / Question 4.
std::cout << "Vui long nhap so Roman: ";
std::string roman;
ex2))) g++ roman_to_int.cpp -o roman_to_int && ./roman_to_int
Vui long nhap so Roman: MCMXCIV
So roman MCMXCIV tuong ung: 1994
ex2))) ./roman_to_int
Vui long nhap so Roman: MCMXC
So roman MCMXC tuong ung: 1990
ex2))) ./roman_to_int
Vui long nhap so Roman: LVIII
So roman LVIII tuong ung: 58
ex2))) ./roman_to_int
Vui long nhap so Roman: LXVI
So roman LXVI tuong ung: 66
ex2))) ./roman_to_int
Vui long nhap so Roman: XXXIX
So roman XXXIX tuong ung: 39
ex2))) rm roman_to_int
ex2)))
TERMINAL
zsh
data_organization
17/14
i
nvim
~/repos/E2E-WEB-PROD
nvim