Câu hỏi 1

Đúng

Đạt điểm 10,00 trên 10,00

Yêu cầu:

Viết hàm `double evaluateFractionString(string fraction)` để tính giá trị của phân số lưu trong 1 xâu ký tự kiểu `string`. Xâu này có định dạng a/b hoặc $-a/b$ với a và b là các số nguyên dương.

Input:

- Hàm nhận vào một string duy nhất.

Output:

- Nếu xâu không tuân thủ định dạng này thì hàm cần ném ngoại lệ với thông điệp `"Bad fraction string."`

Lưu ý:

1. Trong bài này bạn chỉ cần viết hàm `evaluateFractionString`.
2. Chương trình test sẽ đọc dữ liệu từ `stdin`: dòng đầu là số n chỉ định số lượng xâu phân số cần tính giá trị, n dòng tiếp theo mỗi dòng chứa 1 xâu phân số. Chương trình test sẽ in kết quả ra `stdout` tương ứng trên n dòng.

For example:

Test	Input	Result

`int n;`
`cin >> n;`
`cin.ignore();`
`for (int i = 0; i < n; i++) {`
`  string line;`
`  getline(cin, line);`
`  // fflush(stdin);`
`  try {`
`  double val = evaluateFractionString(line);`
`  cout << fixed << setprecision(2) << val << endl;`
`  } catch (const char* str){`
`  cout << "Caught " << str << endl;`
`  }`
`}`
	2
	1/(2
	2/3
		Caught Bad fraction string.
		0.67

Answer: (penalty regime: 0 %)

```
double evaluateFractionString(string fraction) {
  int id = -1;
  int n = fraction.size();
  for (int i = 0; i < n; ++i) {
    if (fraction[i] == '-') {
      continue;
    } else if (fraction[i] == '/') {
      if (id != -1) {
        throw "Bad fraction string.";
      }
      id = i;
    } else if (fraction[i] < '0' || fraction[i] > '9') {
      throw "Bad fraction string.";
    }
  }
  if (id <= 0 || id == n - 1) {
    throw "Bad fraction string.";
  }
  int a = stoi(fraction.substr(0, id));
  int b = stoi(fraction.substr(id + 1));
  if (b == 0) {
    throw "Bad fraction string.";
  }
  return 1.0 * a / b;
}
```

	Test	Input	Expected	Got	

✓	int n;
	cin >> n;
	cin.ignore();
	for (int i = 0; i < n; i++) {
	string line;
	getline(cin, line);
	// fflush(stdin);
	try {
	double val = evaluateFractionString(line);
	cout << fixed << setprecision(2) << val << endl;
	} catch (const char* str){
	cout << "Caught " << str << endl;
	}
	}
		2
		1/(2
		2/3
			Caught Bad fraction string.
			0.67
				Caught Bad fraction string.
				0.67
					✓

Passed all tests! ✓

▸ SHOW/HIDE QUESTION AUTHOR'S SOLUTION (CPP)

Đúng

Marks for this submission: 10,00/10,00.

Câu hỏi 2

Đúng

Đạt điểm 10,00 trên 10,00

Yêu cầu:

Viết chương trình đếm tần suất xuất hiện.

Input:

- Một string là đường dẫn đến 1 tệp văn bản chứa các từ tiếng Anh, mỗi từ 1 dòng.

Output:

- Hãy đọc tệp và in ra từng chữ cái cùng số lần xuất hiện.
- Kết quả in sắp theo thứ tự từ điển.
- Không in những chữ cái không xuất hiện trong tệp.

Gợi ý:

1. Dùng thư viện <map>.

Ví dụ nội dung tệp 1.txt:

deer
duck

For example:

Input	Result
1.txt	c: 1 d: 2 e: 2 k: 1 r: 1 u: 1

Answer: (penalty regime: 0 %)

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

signed main() {
  string path;
  cin >> path;
  map<char, int> cnt;
  ifstream in(path);
  for (string s; in >> s; ) {
    for (char c : s) {
      ++cnt[c];
    }
  }
  for (auto [c, d] : cnt) {
    cout << c << ": " << d << '\n';
  }
}
```

	Input	Expected	Got	
✓	1.txt	c: 1 d: 2 e: 2 k: 1 r: 1 u: 1	c: 1 d: 2 e: 2 k: 1 r: 1 u: 1	✓
✓	2.txt	c: 4 d: 1 e: 4 h: 1 o: 2 p: 2 r: 2 s: 1 u: 1	c: 4 d: 1 e: 4 h: 1 o: 2 p: 2 r: 2 s: 1 u: 1	✓

Passed all tests! ✓

Đúng

Marks for this submission: 10,00/10,00.

Câu hỏi 3

Đúng

Đạt điểm 10,00 trên 10,00

Yêu cầu:

Cài đặt lớp **MyIntSet** biểu diễn tập hợp chứa các giá trị int phân biệt. Lớp sử dụng mảng kích thước cố định để lưu dữ liệu. Biến thành viên num để theo dõi số phần tử đã thêm vào tập hợp. Đề bài đã cung cấp 2 hàm tạo: 1 hàm tạo tập hợp rỗng, 1 hàm tạo tập hợp từ một mảng.

```
class MyIntSet{
public:
    MyIntSet():num(0){}

    MyIntSet(int a[], int n){
        if(n > MAX_SIZE) num = MAX_SIZE;
        else num = n;
        for(int i = 0; i < num; i++) elements[i] = a[i];
    }

    bool insertVal(int v);
    bool eraseVal(int v);
    void clearAll();
    bool findVal(int v) const;
    bool isEmpty() const;
    int getSize() const;
    const int* getBeginPtr() const;
    const int* getEndPtr() const;
private:
    static const int MAX_SIZE = 1000;
    int elements[MAX_SIZE];
    int num; // count real values in this set
};
```

- Hàm `bool insertVal(int v)` thêm giá trị v vào tập hợp. Thêm không thành công (mảng đầy, giá trị v đã có) thì trả về false.
- Hàm `bool eraseVal(int v)` xóa giá trị v khỏi tập hợp. Xóa không thành công (giá trị v chưa có) thì trả về false.
- Hàm `void clearAll()` xóa tất cả các giá trị trong tập hợp.
- Hàm `bool findVal(int v) const` kiểm tra xem v có trong tập hợp không.
- Hàm `bool isEmpty() const` kiểm tra xem tập hợp có rỗng không.
- Hàm `int getSize() const` trả về số phần tử trong tập hợp.
- Hàm `const int* getBeginPtr() const` trả về con trỏ tới phần tử đầu của tập hợp.
- Hàm `const int* getEndPtr() const` trả về con trỏ tới phần tử cuối của tập hợp.

For example:

Test	Result

MyIntSet is1;
is1.eraseVal(6);
is1.insertVal(3);
is1.insertVal(3);
is1.insertVal(5);
vector<int> res;
for(auto p = is1.getBeginPtr(); p <= is1.getEndPtr(); p++){
  res.push_back(*p);
}
sort(res.begin(), res.end());
for(int i = res.size() - 1; i >= 0; i--) cout << res[i] << " ";
	5 3

Answer: (penalty regime: 0 %)

```
1  class MyIntSet {
2  public:
3    MyIntSet() : num(0) {}
4    MyIntSet(int a[], int n) {
5      if (n > MAX_SIZE)
6        num = MAX_SIZE;
```

```
    num = MAX_SIZE;
  else
    num = n;
  for (int i = 0; i < num; i++)
    elements[i] = a[i];
 }
 bool insertVal(int v) {
  if (num == MAX_SIZE || findVal(v)) {
   return false;
  }
  elements[num++] = v;
  return true;
 }
 bool eraseVal(int v) {
  for (int i = 0; i < num; ++i) {
   if (elements[i] == v) {
    elements[i] = elements[--num];
    return true;
   }
  }
  return false;
 }
 void clearAll() { num = 0; }
 bool findVal(int v) const {
  for (int i = 0; i < num; ++i) {
   if (elements[i] == v) {
    return true;
   }
  }
  return false;
 }
 bool isEmpty() const { return num == 0; }
 int getSize() const { return num; }
 const int *getBeginPtr() const { return elements; }
 const int *getEndPtr() const { return elements + num - 1; }

private:
 static const int MAX_SIZE = 1000;
 int elements[MAX_SIZE];
 int num; // count real values in this set
};
```

	Test	Expected	Got	

✓	MyIntSet is1;
	is1.eraseVal(6);
	is1.insertVal(3);
	is1.insertVal(3);
	is1.insertVal(5);
	vector<int> res;
	for(auto p = is1.getBeginPtr(); p <= is1.getEndPtr(); p++){
	  res.push_back(*p);
	}
	sort(res.begin(), res.end());
	for(int i = res.size() - 1; i >= 0; i--) cout << res[i] << " ";
		5 3	5 3	✓

✓	int b[] = {9, 6, 2};
	MyIntSet is1(b, sizeof(b) / sizeof(int));
	is1.eraseVal(6);
	is1.insertVal(3);
	is1.insertVal(3);
	is1.insertVal(5);
	vector<int> res;
	for(auto p = is1.getBeginPtr(); p <= is1.getEndPtr(); p++){
	  res.push_back(*p);
	}
	sort(res.begin(), res.end());
	for(int i = res.size() - 1; i >= 0; i--) cout << res[i] << " ";
		9 5 3 2	9 5 3 2	✓

	Test	Expected	Got	

✓	int b[] = {9, 6, 2};
	MyIntSet is1(b, sizeof(b) / sizeof(int));
	is1.eraseVal(6);
	is1.insertVal(3);
	is1.insertVal(3);
	is1.insertVal(13);
	is1.insertVal(23);
	cout << boolalpha << is1.isEmpty() << endl;
	is1.clearAll();
	cout << boolalpha << is1.isEmpty() << endl;
	is1.insertVal(5);
	is1.eraseVal(3);
	vector<int> res;
	for(auto p = is1.getBeginPtr(); p <= is1.getEndPtr(); p++){
	    res.push_back(*p);
	}
	sort(res.begin(), res.end());
	for(int i = res.size() - 1; i >= 0; i--) cout << res[i] << " ";
		false
		true
		5
			false
			true
			5
				✓

Passed all tests! ✓

Đúng

Marks for this submission: 10,00/10,00.

Câu hỏi **4**

Đúng

Đạt điểm 10,00 trên 10,00

Yêu cầu:

Cài đặt lớp **MyIntSet** biểu diễn tập hợp chứa các giá trị int phân biệt. Lớp sử dụng mảng cấp phát động để lưu dữ liệu. Biến thành viên num để theo dõi số phần tử đã thêm vào tập hợp. Biến thành viên maxSize để theo dõi kích thước tối đa của mảng động.

```
class MyIntSet{
public:
    MyIntSet();
    MyIntSet(int a[], int n);
    ~MyIntSet();
    bool insertVal(int v);
    bool eraseVal(int v);
    void clearAll();
    bool findVal(int v) const;
    bool isEmpty() const;
    int getSize() const;
    const int* getBeginPtr() const;
    const int* getEndPtr() const;
private:
    int maxSize;
    int* elements;
    int num;
};
```

- Hàm `bool insertVal(int v)` thêm giá trị v vào tập hợp. Thêm không thành công (giá trị v đã có) thì trả về false. Mảng đầy thì cấp phát kích thước mới maxSize = maxSize * 2 + 1.
- Hàm `bool eraseVal(int v)` xóa giá trị v khỏi tập hợp. Xóa không thành công (giá trị v chưa có) thì trả về false.
- Hàm `void clearAll()` xóa tất cả các giá trị trong tập hợp.
- Hàm `bool findVal(int v) const` kiểm tra xem v có trong tập hợp không.
- Hàm `bool isEmpty() const` kiểm tra xem tập hợp có rỗng không.
- Hàm `int getSize() const` trả về số phần tử trong tập hợp.
- Hàm `const int* getBeginPtr() const` trả về con trỏ tới phần tử đầu của tập hợp.
- Hàm `const int* getEndPtr() const` trả về con trỏ tới phần tử cuối của tập hợp.

For example:

Test	Result

`MyIntSet is1;`
`is1.eraseVal(6);`
`is1.insertVal(3);`
`is1.insertVal(3);`
`is1.insertVal(5);`
`vector<int> res;`
`for(auto p = is1.getBeginPtr(); p <= is1.getEndPtr(); p++){`
`  res.push_back(*p);`
`}`
`sort(res.begin(), res.end());`
`for(int i = res.size() - 1; i >= 0; i--) cout << res[i] << " ";`
	`5 3`

Answer: (penalty regime: 0 %)

```
class MyIntSet {
public:
  MyIntSet() {
    maxSize = 1000;
    num = 0;
    elements = new int[maxSize];
  }
  MyIntSet(int a[], int n) {
    maxSize = num = n;
    elements = new int[n];
```

```
    for (int i = 0; i < n; ++i) {
      elements[i] = a[i];
    }
  }
  ~MyIntSet() {
    delete[] elements;
    maxSize = num = 0;
  }
  bool insertVal(int v) {
    if (findVal(v)) {
      return false;
    }
    if (num == maxSize) {
      int *tmp = new int[2 * num + 1];
      for (int i = 0; i < num; ++i) {
        tmp[i] = elements[i];
      }
      maxSize = 2 * maxSize + 1;
      swap(tmp, elements);
      delete[] tmp;
    }
    elements[num++] = v;
    return true;
  }
  bool eraseVal(int v) {
    for (int i = 0; i < num; ++i) {
      if (elements[i] == v) {
        elements[i] = elements[--num];
        return true;
      }
    }
    return false;
  }
  void clearAll() {
    num = 0;
  }
  bool findVal(int v) const {
    for (int i = 0; i < num; ++i) {
      if (elements[i] == v) {
        return true;
      }
    }
```

	Test	Expected	Got	

✓	MyIntSet is1;
	is1.eraseVal(6);
	is1.insertVal(3);
	is1.insertVal(3);
	is1.insertVal(5);
	vector<int> res;
	for(auto p = is1.getBeginPtr(); p <= is1.getEndPtr(); p++){
	  res.push_back(*p);
	}
	sort(res.begin(), res.end());
	for(int i = res.size() - 1; i >= 0; i--) cout << res[i] << " ";
		5 3	5 3	✓

✓	int b[] = {9, 6, 2};
	MyIntSet is1(b, sizeof(b) / sizeof(int));
	is1.eraseVal(6);
	is1.insertVal(3);
	is1.insertVal(3);
	is1.insertVal(5);
	vector<int> res;
	for(auto p = is1.getBeginPtr(); p <= is1.getEndPtr(); p++){
	  res.push_back(*p);
	}
	sort(res.begin(), res.end());
	for(int i = res.size() - 1; i >= 0; i--) cout << res[i] << " ";
		9 5 3 2	9 5 3 2	✓

	Test	Expected	Got	

✓	int b[] = {9, 6, 2};
	MyIntSet is1(b, sizeof(b) / sizeof(int));
	is1.eraseVal(6);
	is1.insertVal(3);
	is1.insertVal(3);
	is1.insertVal(13);
	is1.insertVal(23);
	cout << boolalpha << is1.isEmpty() << endl;
	is1.clearAll();
	cout << boolalpha << is1.isEmpty() << endl;
	is1.insertVal(5);
	is1.eraseVal(3);
	vector<int> res;
	for(auto p = is1.getBeginPtr(); p <= is1.getEndPtr(); p++){
	  res.push_back(*p);
	}
	sort(res.begin(), res.end());
	for(int i = res.size() - 1; i >= 0; i--) cout << res[i] << " ";
		false
		true
		5
			false
			true
			5
				✓

Passed all tests! ✓

Đúng

Marks for this submission: 10,00/10,00.

Trở lại Khoá học