ΟΝΟΜΑ: Μηλτιάδης
ΕΠΩΝΥΜΟ: Μαντές
ΑΜ: 1084661
ΕΤΟΣ: 2

10

ΑΣΚΗΣΗ ΕΡΓΑΣΤΗΡΙΟΥ: 2

- **ΣΧΟΛΙΑΣΜΟΣ ΛΥΣΗΣ**

Αρχικά, ορίζουμε μια συνάρτηση MergeSortedVectors() με ορίσματα δύο διανύσματα <int> και τύπο επιστροφής πάλι διάνυσμα <int>. Μέσα στο σώμα της συνάρτησης για να κάνουμε τη συγχώνευση των δύο πινάκων κάνουμε το εξής μέχρι οι μετρητές i, j να φτάσουν στο τέρμα των δύο πινάκων: Αν το i - στο στοιχείο του v1 είναι μικρότερο από το j - στο στοιχείο του v2 τότε με τη συνάρτηση push_back() αποθηκεύουμε στο διάνυσμα v3 που ορίσαμε το στοιχείο αυτό και αυξάνουμε τη τιμή του i κατά 1 για να οδηγηθούμε στο επόμενο στοιχείο του v1. Την αντίστοιχη διαδικασία ακολουθούμε και αν το i - στο στοιχείο του v1 είναι μεγαλύτερο από το j - στο στοιχείο του v2 μόνο που τότε αυξάνουμε τον μετρητή j και αποθηκεύουμε στο διάνυσμα v3 το στοιχείο του v2. Τέλος, για να ελέγξουμε και τη περίπτωση που τα δύο διανύσματα έχουν διαφορετικό μέγεθος, πάλι με ένα βρόχο while αποθηκεύουμε στο v3 όσα στοιχεία περίσσεψαν στο v1 ή στο v2. Αφού ολοκληρωθεί η συγχώνευση των δύο πινάκων στο v3 επιστρέφουμε το διάνυσμα v3 με την εντολή return. Έπειτα, ορίζουμε τη συνάρτηση binarySearch() με ορίσματα ένα διάνυσμα <int>, και τρεις ακεραίους low, high, toSearch που εκφράζουν τη μικρότερη και τη μεγαλύτερη θέση του διανύσματος καθώς και το στοιχείο που θέλουμε να ψάξουμε. Ως τύπο επιστροφής ορίζουμε και πάλι το int. Μέσα στο σώμα της συνάρτησης ελέγχουμε αν η τιμή του high είναι μεγαλύτερη ή ίση του low για να εξασφαλίσουμε ότι το στοιχείο που ψάχνουμε υπάρχει στο πίνακα και αν ναι υπολογίζουμε τη μεσαία θέση κάθε φορά. Αν το στοιχείο του πίνακα στη μεσαία θέση είναι το ζητούμενο τότε γίνεται επιστροφή της τιμής true (1), ενώ αν είναι μεγαλύτερο από το ζητούμενο συνεχίζεται η αναζήτηση προς τα αριστερά του πίνακα ή αν είναι μικρότερο προς τα δεξιά, με αναδρομή, μέχρι να βρεθεί. Σε περίπτωση που δε βρεθεί επιστρέφουμε τιμή false (0). Μέσα στη main() διαβάζουμε τους αριθμούς από τα δύο αρχεία και τους αποθηκεύουμε σε δύο διανύσματα array_1, array_2. Ωστόσο, επειδή τα στοιχεία στο δεύτερο αρχείο είναι σε φθίνουσα σειρά, τα κατατάσσουμε σε αύξουσα κάνοντας το εξής : Ορίζουμε μια μεταβλητή temp και αν το προηγούμενο στοιχείο του array_2 είναι μεγαλύτερο από το

επόμενο, τότε το εκχωρούμε σε αυτήν. Μετά, στη θέση του προηγούμενου στοιχείου εκχωρούμε το επόμενο και τη τιμή του προηγούμενου που την έχουμε αποθηκεύσει στην temp την εκχωρούμε στη θέση του επόμενου. Τέλος, καλούμε τη MergeSortedVectors() με ορίσματα τα διανύσματα array_1, array_2 και το συγχωνευμένο διάνυσμα που επιστρέφει το εκχωρούμε στο mergedarray. Ταυτόχρονα, ζητάμε και από το χρήστη να εισάγει έναν αριθμό που θέλει να ψάξει στον συγχωνευμένο πίνακα και έτσι καλούμε τη binarySearch() με ορίσματα το διάνυσμα mergedarray και τις τιμές 0, array_1.size() + array_2.size() - 1 και num. Το αποτέλεσμα (0 ή 1) αν βρέθηκε το στοιχείο το εκχωρούμε σε μια μεταβλητή result και ανάλογα με τη τιμή του εκτυπώνουμε και το αντίστοιχο μήνυμα.

- **ΚΩΔΙΚΑΣ ΛΥΣΗΣ**

```
#include <iostream>
#include <fstream>
#include <vector>

using namespace std;

vector<int> MergeSortedVectors(vector<int>& v1,
vector<int>& v2)
{
     vector<int> v3;

     int i, j, n, m;
     i = j = 0;
     n = v1.size();
     m = v2.size();

     while (i < n && j < m)
    {
          if (v1[i] <= v2[j])
          {
```

```
                v3.push_back(v1[i]);
                ++i;
            }
            else
            {
                v3.push_back(v2[j]);
                ++j;
            }
        }

        while (i < n) {
            v3.push_back(v1[i]);
            ++i;
        }

        while (j < m) {
            v3.push_back(v2[j]);
            ++j;
        }

        return v3;
}

int binarySearch(vector<int> &arr, int low, int high, int
toSearch)
    {
        if (high >= low) {
            int mid = (high + low) / 2;

            if (arr[mid] == toSearch) return 1;
```

```
            else if (arr[mid] > toSearch) return
binarySearch(arr, low, mid - 1, toSearch);


            else return binarySearch(arr, mid + 1, high,
toSearch);
        }
        return 0;
    }


int main()
{
     vector<int> array_1, array_2;
     vector<int> mergedarray;
     int current_number_1, current_number_2 = 0;
     int m, n, temp;


    ifstream input_file_1("Array1.txt");
    ifstream input_file_2("Array2.txt");


    if (input_file_1.good())
     {
        while (input_file_1 >> current_number_1)
array_1.push_back(current_number_1);
        input_file_1.close();


        // Display the numbers read:
        cout << "The numbers of the 1st array are: ";
        for (int count = 0; count < array_1.size();
count++){
            cout << array_1[count] << " ";
        }
        cout << endl;
```

```
    }else {
        cout << "Error in reading file!";
        exit(0);
    }

    if (input_file_2.good())
     {
        while (input_file_2 >> current_number_2)
array_2.push_back(current_number_2);
        input_file_2.close();

        for(n = 0; n < array_2.size(); n++)
        {
            for(m = n + 1; m < array_2.size(); m++)
            {
                if(array_2[n] > array_2[m])
                {
                    temp = array_2[n];
                    array_2[n] = array_2[m];
                    array_2[m] = temp;
                }
            }
        }

        // Display the numbers read:
        cout << "\nThe numbers of the 2nd array are: ";
        for (int count = 0; count < array_2.size();
count++) cout << array_2[count]<< " ";
        cout << endl;
```

```
    }else {
        cout << "Error in reading file!";
        exit(0);
    }

    mergedarray = MergeSortedVectors(array_1, array_2);

    // Display the numbers read:
    cout << "\nThe numbers of the merged array are: ";
    for (int count = 0; count < mergedarray.size();
count++) cout << mergedarray[count]<< " ";
    cout << endl;

    int num;
    cout << "\n\nEnter the integer of the merged array
you want to search for : ";
    cin >> num;

    int low = 0;
    int high = array_1.size() + array_2.size() - 1;

    int result = binarySearch(mergedarray, low, high,
num);
    if (result == 1) cout << "\nElement is found";
    else cout << "\nElement is not found";
}
```

- **ΑΠΟΤΕΛΕΣΜΑΤΑ ΕΚΤΕΛΕΣΗΣ**

```
The numbers of the 1st array are: 2 3 6 9 38 57 68 73 74
102 121 136 138 146 150 183 185 197 248 249 270
273 280 289 309 322 326 334 336 339 342 352 362 376 380
382 388 393 398 430 438 443 460 463 466 470 476
```

477 521 526 544 552 563 578 592 596 610 644 646 648 650
651 656 659 675 694 700 736 738 748 755 755 761

762 765 788 791 803 821 838 841 843 844 844 849 856 856
863 863 866 882 887 894 915 915 916 926 936 949 985

The numbers of the 2nd array are: 6 7 9 20 23 44 50 66 81
86 95 99 108 121 141 153 164 166 171 215 229 229

234 234 240 247 258 259 268 284 287 288 291 307 310 320
324 352 353 362 362 372 383 399 399 412 453 456 465

473 477 483 485 499 509 518 522 522 524 532 546 550 573
580 584 592 614 619 630 635 638 638 646 661 664 678

730 735 750 764 792 819 842 845 845 854 856 872 893 904
915 930 941 954 958 970 972 983 987 997

The numbers of the merged array are: 2 3 6 6 7 9 9 20 23
38 44 50 57 66 68 73 74 81 86 95 99 102 108 121 121

136 138 141 146 150 153 164 166 171 183 185 197 215 229
229 234 234 240 247 248 249 258 259 268 270 273 280

284 287 288 289 291 307 309 310 320 322 324 326 334 336
339 342 352 352 353 362 362 362 372 376 380 382 383

388 393 398 399 399 412 430 438 443 453 456 460 463 465
466 470 473 476 477 477 483 485 499 509 518 521 522

522 524 526 532 544 546 550 552 563 573 578 580 584 592
592 596 610 614 619 630 635 638 638 644 646 646 648

650 651 656 659 661 664 675 678 694 700 730 735 736 738
748 750 755 755 761 762 764 765 788 791 792 803 819

821 838 841 842 843 844 844 845 845 849 854 856 856 856
863 863 866 872 882 887 893 894 904 915 915 915 916

926 930 936 941 949 954 958 970 972 983 985 987 997

Enter the integer of the merged array you want to search
for : 66

Element is found