**Zadaci za vježbu**

Contents

[Napomena (obavezno procitati): 1](#_Toc155550595)

[1. Zadaci sa vjezbi 2](#_Toc155550596)

[2. Rekurzije 30](#_Toc155550597)

[3. Zadaci za pripremu 34](#_Toc155550598)

# Napomena (obavezno procitati):

Dokument sadrži primjere zadataka za vježbu. Dokument će biti proširen u narednom periodu. Dokument se ne smatra zbirkom zadataka i ni na koji način ne predstavlja listu potencijalnih ispitnih zadataka, niti obavezuje da će ispitni zadaci biti isti ili slični onima navedenim u ovom dokumentu. Dokument je namjenjen isključivo studentima prve godine i nije ga dozvoljeno dijeliti bez odobrenja profesora zaduženog za predmet.

U zadacima sa vježbi, mogu se naći rješenja **koja nisu u potpunosti optimizirana**. Neoptimizirana rjesenja se nekada koriste na vježbama zbog lakšeg razumjevanja od strane studenata koji su početnici u programiranju. Također, princip kreiranja odvojenih funkcija ne predstavlja najoptimalniji pristup u osmišljavanju kompletnog rješenja. Npr. funkcija koja samo vraća random broj, funkcija koja provjerava uvjet koji se moze uraditi samo jednom linijom code-a, odnosno samo jednim izrazom ili bilo koja druga funkcija koja ne pruža mogućnost koja bi programu dala neku buduću univerzalniju funkcionalnost ne treba biti odvojena. Na vježbama je to dozvoljeno zbog lakšeg razumjevanja strukture složenih funkcionalnosti, dok će na ispitima biti traženo da se ne kreiraju funkcije za koje nema potrebe. U slučaju da se određene kompleksnije funkcionalnosti code-a mogu odvojiti u zasebne blokove code-a, na ispitu će biti poželjno da se kreiraju i odvojene funkcije za te blokove.

# 1. Zadaci sa vjezbi

=================================================== POSTAVKA 1

Napravite rekurzivnu funkciju ‘nthFibonacci' za računanje n - tog Fibonacci broja.

=================================================== CODE 1

#include <iostream>

using namespace std;

int nthFibonacci(int n) {

if (n <= 2) return 1;

return nthFibonacci(n - 1) + nthFibonacci(n - 2);

}

int main() {

int n;

do

{

cout << "Unesite 'n': " << endl;

cin >> n;

} while (n < 1);

cout << "Odgovarajuci Fibonaccijev broj je " << nthFibonacci(n) << endl;

return 0;

}

=================================================== POSTAVKA 2

Napravite rekurzivnu funkciju 'printLettersInReverse', koja treba da ispiše sva velika slova koja se nalaze između dva proslijeđena velika slova (ali u obratnom redoslijedu).

=================================================== CODE 2

#include <iostream>

using namespace std;

void printLettersBetweenInReverse(char s1, char s2) {

if (!isupper(s1) || !isupper(s2)) return;

if (s1 > s2) return;

printLettersBetweenInReverse(char(s1 + 1), s2);

cout << s1;

}

int main() {

char slovo1, slovo2;

do

{

do{

cout << "Unesite prvo slovo:" << endl;

cin >> slovo1;

} while (!isupper(slovo1));

do {

cout << "Unesite drugo slovo:" << endl;

cin >> slovo2;

} while (!isupper(slovo2));

} while (slovo1 > slovo2);

cout << "Ispis slova izmedju dva unesena: " << endl;

printLettersBetweenInReverse(slovo1, slovo2);

return 0;

}

=================================================== POSTAVKA 3

Napišite program koji će omogućiti kreiranje 1D niza proizvoljne veličine, te uz pomoć rekurzivnih funkcija omogućiti:

-> unos elemenata

-> ispis elemenata

-> izračunati sumu svih elemenata niza

-> sumu pozitivnih elemenata niza

-> sumu elemenata niza s parnim indeksom

=================================================== CODE 3

#include <iostream>

using namespace std;

const int MAX\_SIZE = 100;

void enterArray(int arr[], int i, int size) {

if (i == size)

return;

cout << "Unesite element na poziciji [" << i << "]:";

cin >> arr[i];

enterArray(arr, i + 1, size);

}

void printArray(int arr[], int i, int size) {

if (i == size)

return;

cout << arr[i] << " ";

printArray(arr, i + 1, size);

}

int sumOfArray(int arr[], int i, int size) {

if (i == size)

return 0;

return arr[i] + sumOfArray(arr, i + 1, size);

}

int positiveSumOfArray(int arr[], int i, int size) {

if (i == size)

return 0;

return (arr[i] > 0 ? arr[i] : 0) + positiveSumOfArray(arr, i + 1, size);

}

int sumOfEvenIndices(int arr[], int i, int size) {

if (i == size) return 0;

return (i % 2 == 0 ? arr[i] : 0) + sumOfEvenIndices(arr, i + 1, size);

}

int main() {

int n;

int arr[MAX\_SIZE];

do{

cout << "Unesite velicinu niza: " << endl;

cin >> n;

} while (n < 1);

enterArray(arr, 0, n);

printArray(arr, 0, n);

cout << "Suma svih elemenata u nizu je " << sumOfArray(arr, 0, n) << endl;

cout << "Suma svih pozitivnih elemenata u nizu je " << positiveSumOfArray(arr, 0, n) << endl;

cout << "Suma elemenata sa parnim indeksima u nizu je " << sumOfEvenIndices(arr, 0, n) << endl;

return 0;

}

=================================================== POSTAVKA 4

Napišite rekurzivnu funkciju 'isPrime', koja provjerava da li je dati broj prost.

=================================================== CODE 4

#include <iostream>

using namespace std;

bool isPrime(int number, int i = 2) {

if (number <= 1) return false;

if (number == 2 || number == 3) return true;

if (number % i == 0) return false;

if (i == number - 1) return true;

isPrime(number, i + 1);

}

int main() {

int n;

cout << "Unesite broj 'n': " << endl;

cin >> n;

cout << "Broj " << (isPrime(n) ? "je" : "nije") << " prost." << endl;

return 0;

}

=================================================== POSTAVKA 5

Napisati program za sabiranje matrica proizvoljnih dimenzija.

Neka su date dvije matrice A i B, koje je potrebno popuniti slučajnim vrijednostima u opsegu [1-5]

Formirati matricu C kao zbir matrica A i B:

C[i,j] = A[i,j] + B[i,j]

CODE 5 =====================================================================

#include <iostream>

#include <iomanip>

using namespace std;

int getRandomValue(int from, int to) {

return rand() % (to - from + 1) + from;

}

// funkcija za kreiranje dinamičke matrice

int\*\* createMatrix(int rows, int columns) {

int\*\* m = new int\* [rows];

for (int i = 0; i < rows; i++)

m[i] = new int[columns] {0}; // inicijalizacija dinamičkih varijabli na 0

return m;

}

// funkcija za setovanje random vrijednosti

void setRandomValues(int\*\* matrix, int rows, int columns, int from, int to) {

for (int i = 0; i < rows; i++)

for (int j = 0; j < columns; j++)

matrix[i][j] = getRandomValue(from, to);

}

// funkcija za printanje matrice

void printMatrix(int\*\* matrix, int rows, int columns) {

for (int i = 0; i < rows; i++) {

for (int j = 0; j < columns; j++)

cout << setw(3) << matrix[i][j] << " ";

cout << endl;

}

}

// funkcija za dealokaciju matrice

void deleteMatrix(int\*\* matrix, int rows) {

for (int i = 0; i < rows; i++) {

delete[] matrix[i];

matrix[i] = nullptr;

}

delete[] matrix;

}

int\*\* addmatrices(int\*\* m1, int\*\* m2, int rows, int columns) {

int\*\* m3 = createMatrix(rows, columns);

for (int i = 0; i < rows; i++)

for (int j = 0; j < columns; j++)

m3[i][j] = m1[i][j] + m2[i][j];

return m3;

}

int main() {

srand(42);

int rows = 1, columns = 1;

do

{

cout << "Unesite broj redova: " << endl;

cin >> rows;

cout << "Unesite broj kolona: " << endl;

cin >> columns;

} while (rows < 1 || columns < 1);

// kreiranje matrica m1 i m2

int\*\* m1 = createMatrix(rows, columns);

int\*\* m2 = createMatrix(rows, columns);

// randomizacija vrijednosti

int from = 1, to = 5;

setRandomValues(m1, rows, columns, from, to);

setRandomValues(m2, rows, columns, from, to);

// printanje prve matrice

cout << "Matrica 1: " << endl;

printMatrix(m1, rows, columns);

// printanje druge matrice

cout << "Matrica 2: " << endl;

printMatrix(m2, rows, columns);

int\*\* m3 = addmatrices(m1, m2, rows, columns);

cout << "Zbirna matrica: " << endl;

printMatrix(m3, rows, columns);

// dealokacija matrica

deleteMatrix(m1, rows);

deleteMatrix(m2, rows);

deleteMatrix(m3, rows);

return 0;

}

=================================================== POSTAVKA 6

Neka je data matrica M proizvoljnih dimenzija koje određuje korisnik.

Popuniti matricu M slučajnim vrijednostima u opsegu [6-10]

Napisati dvije funkcije:

-> getAveragesByRow [funkcija koja treba da vrati prosjeke redova]

-> getAveragesByColumn [funkcija koja treba da vrati prosjeke kolona

CODE 6 =====================================================================

#include <iostream>

#include <iomanip>

using namespace std;

int getRandomValue(int from, int to) {

return rand() % (to - from + 1) + from;

}

int\*\* createMatrix(int rows, int columns) {

int\*\* m = new int\* [rows];

for (int i = 0; i < rows; i++)

m[i] = new int[columns] {0}; // inicijalizacija dinamičkih varijabli na 0

return m;

}

void setRandomValues(int\*\* matrix, int rows, int columns, int from, int to) {

for (int i = 0; i < rows; i++)

for (int j = 0; j < columns; j++)

matrix[i][j] = getRandomValue(from, to);

}

void printMatrix(int\*\* matrix, int rows, int columns) {

for (int i = 0; i < rows; i++) {

for (int j = 0; j < columns; j++)

cout << setw(3) << matrix[i][j] << " ";

cout << endl;

}

}

void deleteMatrix(int\*\* matrix, int rows) {

for (int i = 0; i < rows; i++) {

delete[] matrix[i];

matrix[i] = nullptr;

}

delete[] matrix;

}

// funkcija za kreiranje dinamičkog float niza

float\* createArray(int size) {

return new float[size] {0.0f}; // inicijalizacija svih vrijednosti niza na 0.0f

}

void printArray(float\* p, int size, const char\* delimiter = " ") {

for (int i = 0; i < size; i++)

cout << p[i] << delimiter;

cout << endl;

}

void deleteArray(float\* p) {

delete[] p;

}

float\* getAveragesByRow(int\*\* m, int rows, int columns) {

float\* averages = createArray(rows); // kreiranje niza za pohranu prosjeka po redovima

for (int i = 0; i < rows; i++)

{

for (int j = 0; j < columns; j++)

averages[i] += m[i][j];

averages[i] /= columns;

}

return averages;

}

float\* getAveragesByColumn(int\*\* m, int rows, int columns) {

float\* averages = createArray(columns); // kreiranje niza za pohranu prosjeka po kolona

for (int j = 0; j < columns; j++)

{

for (int i = 0; i < rows; i++)

averages[j] += m[i][j];

averages[j] /= rows;

}

return averages;

}

int main() {

srand(42);

int rows = 1, columns = 1;

do

{

cout << "Unesite broj redova: " << endl;

cin >> rows;

cout << "Unesite broj kolona: " << endl;

cin >> columns;

} while (rows < 1 || columns < 1);

// kreiranje matrice m1

int\*\* m1 = createMatrix(rows, columns);

// randomizacija vrijednosti

int from = 6, to = 10;

setRandomValues(m1, rows, columns, from, to);

// printanje matrice

printMatrix(m1, rows, columns);

// računanje prosjeka po redovima

cout << "Prosjeci po redovima matrice: " << endl;

float\* rowAverages = getAveragesByRow(m1, rows, columns);

printArray(rowAverages, rows);

// računanje prosjeka po kolonama

cout << "Prosjeci po kolonama matrice: " << endl;

float\* columnAverages = getAveragesByColumn(m1, rows, columns);

printArray(columnAverages, columns);

// dealokacija

deleteMatrix(m1, rows);

deleteArray(rowAverages);

deleteArray(columnAverages);

return 0;

}

=================================================== POSTAVKA 7

Napisati program koji omogućuje korisniku da unese dimenzije dviju matrica M1 i M2.

Popuniti matrice M1 i M2 slučajnim vrijednostima u opsegu [1-20]

Napisati funkciju:

-> hstack [funkcija koja prima kao ulazne parametre dvije matrice M1 i M2 a vraća novu matricu M3

koja se formira horizontalnim slaganjem matrica M1 i M2

Primjer:

Neka je M1:

1 2 3

4 5 6

7 8 9

Neka je M2:

4 10

7 8

1 1

Rezultujuća matrica M3 treba igledati ovako:

1 2 3 4 10

4 5 6 7 8

7 8 9 1 1

CODE 7 =====================================================================

#include <iostream>

#include <iomanip>

using namespace std;

int getRandomValue(int from, int to) {

return rand() % (to - from + 1) + from;

}

int\*\* createMatrix(int rows, int columns) {

int\*\* m = new int\* [rows];

for (int i = 0; i < rows; i++)

m[i] = new int[columns] {0}; // inicijalizacija dinamičkih varijabli na 0

return m;

}

void setRandomValues(int\*\* matrix, int rows, int columns, int from, int to) {

for (int i = 0; i < rows; i++)

for (int j = 0; j < columns; j++)

matrix[i][j] = getRandomValue(from, to);

}

void printMatrix(int\*\* matrix, int rows, int columns) {

for (int i = 0; i < rows; i++) {

for (int j = 0; j < columns; j++)

cout << setw(3) << matrix[i][j] << " ";

cout << endl;

}

}

void deleteMatrix(int\*\* matrix, int rows) {

for (int i = 0; i < rows; i++) {

delete[] matrix[i];

matrix[i] = nullptr;

}

delete[] matrix;

}

int\*\* hstack(int\*\* m1, int\*\* m2, int r1, int c1, int r2, int c2) {

if (r1 != r2) {

cout << "Matrice imaju razlicit broj redova, pa prema tome - horizontalno slaganje se ne moze izvesti.";

return nullptr; // u nedostatku drugih mogućnosti vraćamo nultu adresu

}

int\*\* m3 = createMatrix(r1, c1 + c2);

int columnCounter = 0;

for (int j = 0; j < c1; j++) {

for (int i = 0; i < r1; i++)

m3[i][columnCounter] = m1[i][j];

columnCounter++;

}

for (int j = 0; j < c2; j++) {

for (int i = 0; i < r1; i++)

m3[i][columnCounter] = m2[i][j];

columnCounter++;

}

return m3;

}

int main() {

srand(42);

int r1 = 1, c1 = 1, r2 = 1, c2 = 2;

do{

cout << "Unesite broj redova za prvu matricu: " << endl;

cin >> r1;

cout << "Unesite broj kolona za prvu matricu: " << endl;

cin >> c1;

} while (r1 < 1 || c1 < 1);

do {

cout << "Unesite broj redova za drugu matricu: " << endl;

cin >> r2;

cout << "Unesite broj kolona za drugu matricu: " << endl;

cin >> c2;

} while (r2 < 1 || c2 < 1);

// kreiranje matrice m1

int\*\* m1 = createMatrix(r1, c1);

// kreiranje matrice m2

int\*\* m2 = createMatrix(r2, c2);

// randomizacija vrijednosti

int from = 1, to = 20;

setRandomValues(m1, r1, c1, from, to);

setRandomValues(m2, r2, c2, from, to);

// printanje prve matrice

cout << "Matrica M1: " << endl;

printMatrix(m1, r1, c1);

cout << endl;

cout << "Matrica M2: " << endl;

printMatrix(m2, r2, c2);

// horizontalno slaganje matrica

int\*\* m3 = hstack(m1, m2, r1, c1, r2, c2);

int r3 = r1, c3 = c1 + c2;

if (m3 != nullptr) {

cout << "Matrica M3 formirana horizontalnim slaganjem matrica M1 i M2: " << endl;

printMatrix(m3, r3, c3);

}

// dealokacija

deleteMatrix(m1, r1);

deleteMatrix(m2, r2);

deleteMatrix(m3, r3);

return 0;

}

=================================================== POSTAVKA 8

Napisati program koji omogućuje korisniku da unese dimenzije dviju matrica M1 i M2.

Popuniti matrice M1 i M2 slučajnim vrijednostima u opsegu [1-20]

Napisati funkciju:

-> vstack [funkcija koja prima kao ulazne parametre dvije matrice M1 i M2 a vraća novu matricu M3

koja se formira vertikalnim slaganjem matrica M1 i M2

Primjer:

Neka je M1:

1 2 3

4 5 6

7 8 9

Neka je M2:

4 10 5

1 1 2

Rezultujuća matrica M3 treba igledati ovako:

1 2 3

4 5 6

7 8 9

4 10 5

1 1 2

CODE 8 ===================================================================

#include <iostream>

#include <iomanip>

using namespace std;

int getRandomValue(int from, int to) {

return rand() % (to - from + 1) + from;

}

int\*\* createMatrix(int rows, int columns) {

int\*\* m = new int\* [rows];

for (int i = 0; i < rows; i++)

m[i] = new int[columns] {0}; // inicijalizacija dinamičkih varijabli na 0

return m;

}

void setRandomValues(int\*\* matrix, int rows, int columns, int from, int to) {

for (int i = 0; i < rows; i++)

for (int j = 0; j < columns; j++)

matrix[i][j] = getRandomValue(from, to);

}

void printMatrix(int\*\* matrix, int rows, int columns) {

for (int i = 0; i < rows; i++) {

for (int j = 0; j < columns; j++)

cout << setw(3) << matrix[i][j] << " ";

cout << endl;

}

}

void deleteMatrix(int\*\* matrix, int rows) {

for (int i = 0; i < rows; i++) {

delete[] matrix[i];

matrix[i] = nullptr;

}

delete[] matrix;

}

int\*\* vstack(int\*\* m1, int\*\* m2, int r1, int c1, int r2, int c2) {

if (c1 != c2) {

cout << "Matrice imaju razlicit broj kolona, pa prema tome - vertikalno slaganje se ne moze izvesti.";

return nullptr; // u nedostatku drugih mogućnosti vraćamo nultu adresu

}

int\*\* m3 = createMatrix(r1 + r2, c1);

int rowCounter = 0;

for (int i = 0; i < r1; i++){

for (int j = 0; j < c1; j++)

m3[rowCounter][j] = m1[i][j];

rowCounter++;

}

for (int i = 0; i < r2; i++){

for (int j = 0; j < c2; j++)

m3[rowCounter][j] = m2[i][j];

rowCounter++;

}

return m3;

}

int main() {

srand(42);

int r1 = 1, c1 = 1, r2 = 1, c2 = 2;

do{

cout << "Unesite broj redova za prvu matricu: " << endl;

cin >> r1;

cout << "Unesite broj kolona za prvu matricu: " << endl;

cin >> c1;

} while (r1 < 1 || c1 < 1);

do {

cout << "Unesite broj redova za drugu matricu: " << endl;

cin >> r2;

cout << "Unesite broj kolona za drugu matricu: " << endl;

cin >> c2;

} while (r2 < 1 || c2 < 1);

// kreiranje matrice m1

int\*\* m1 = createMatrix(r1, c1);

// kreiranje matrice m2

int\*\* m2 = createMatrix(r2, c2);

// randomizacija vrijednosti

int from = 1, to = 20;

setRandomValues(m1, r1, c1, from, to);

setRandomValues(m2, r2, c2, from, to);

// printanje prve matrice

cout << "Matrica M1: " << endl;

printMatrix(m1, r1, c1);

cout << endl;

cout << "Matrica M2: " << endl;

printMatrix(m2, r2, c2);

// vertikalno slaganje matrica

int\*\* m3 = vstack(m1, m2, r1, c1, r2, c2);

int r3 = r1 + r2, c3 = c1;

if (m3 != nullptr) {

cout << "Matrica M3 formirana vertikalnim slaganjem matrica M1 i M2: " << endl;

printMatrix(m3, r3, c3);

}

// dealokacija

deleteMatrix(m1, r1);

deleteMatrix(m2, r2);

deleteMatrix(m3, r3);

return 0;

}

POSTAVKA 9 ==========================================================================

Kreirati strukturu Datum sa tri obilježja (dan, mjesec i godina) a zatim implementirati program

gdje je potrebno unijeti niz datuma proizvoljne veličine (veličine koju određuje korisnik), nakon čega će program ispisati najstariji datum od unesenih.

Koristiti sljedeći prototip funkcije za pronalaženje najstarijeg datuma:

Datum\* getOldestDate(Datum\* d, int size);

Napomene: Uneseni datumi moraju biti validni datumi [u suprotnom ponavljati unos]

Za provjeru validnosti datuma koristite funkciju članicu 'isValid'

Obavezno uraditi dealokaciju nizu datuma na kraju programa.

CODE 9 ==================================================================

#include<iostream>

#include<iomanip>

using namespace std;

bool isLeap(int godina) {

return (godina % 4 == 0 && godina % 100 != 0) || godina % 400 == 0;

}

int getDaysForMonth(int mjesec, int godina) {

switch (mjesec)

{

case 4: case 6: case 9: case 11: return 30;

case 2:

return isLeap(godina) ? 29 : 28;

default:

return 31;

}

}

struct Datum

{

int dan;

int mjesec;

int godina;

void unos() {

cout << "Unesite dan: " << endl;

cin >> dan;

cout << "Unesite mjesec: " << endl;

cin >> mjesec;

cout << "Unesite godinu: " << endl;

cin >> godina;

}

void ispis() {

cout << dan << "." << mjesec << "." << godina << endl;

}

bool isValid() {

if (dan < 1 || dan > 31 || mjesec < 1 || mjesec > 12 || godina < 1)

return false;

return dan <= getDaysForMonth(mjesec, godina);

}

int toInt() {

return 365 \* godina + 30 \* mjesec + dan;

}

};

// koristimo pokazivačke varijable za rad sa adresama objekata iz niza

// povratna vrijednost je adresa objekta koji ima manju "datumsku" vrijednost

Datum\* min(Datum\* d1, Datum\* d2) {

return (d1->toInt() < d2->toInt()) ? d1 : d2;

}

Datum\* getOldestDate(Datum \* d, int size) {

Datum\* temp = d; // kreiranje pokazivača i setovanje istog na adresu prvog elementa

for (int i = 0; i < size; i++)

temp = min(temp, d + i);

return temp;

}

int main() {

int vel;

do

{

cout << "Unesite dimenzije niza datuma: " << endl;

cin >> vel;

} while (vel < 1);

// kreiranje dinamičkog niza datuma

Datum\* d = new Datum[vel];

for (int i = 0; i < vel; i++)

{

do

{

cout << "--------------------------------------\n";

cout << "Unesite datum na lokaciji " << i <<":"<< endl;

d[i].unos(); // alternativno: \*(d+i).unos();

} while (!d[i].isValid()); // ponavljanje unosa datuma d[i] ako je d[i] nevalidan datum

}

for (int i = 0; i < vel; i++)

{

cout << "Datum " << i << ": ";

d[i].ispis();

}

// dohvaćanje adresa objekta sa najmanjom "datumskom" vrijednošću atributa (dan,mjesec,godina)

Datum\* oldest = getOldestDate(d, vel);

cout << "Najstariji datum od unesenih je: ";

oldest->ispis();

// dealokacija dinamičkog niza datuma

delete[] d;

d = nullptr;

return 0;

}

POSTAVKA 10=======================================================================

Kreirati strukturu Datum sa tri obilježja (dan, mjesec i godina) a zatim implementirati program

gdje je potrebno generisati niz slučajnih (random) datuma proizvoljne veličine (veličine koju određuje korisnik), nakon čega će program ispisati najnoviji datum od unesenih.

Koristiti sljedeći prototip funkcije za pronalaženje najstarijeg datuma:

Datum\* getNewestDate(Datum\* d, int size);

Napomene: Generisani slučajni datumi moraju biti validni datumi [u suprotnom ponavljati generisanje]

Za provjeru validnosti datuma koristite funkciju članicu 'isValid'

Obavezno uraditi dealokaciju nizu datuma na kraju programa.

CODE 10=================================================================

#include<iostream>

#include<iomanip>

using namespace std;

bool isLeap(int godina) {

return (godina % 4 == 0 && godina % 100 != 0) || godina % 400 == 0;

}

int getDaysForMonth(int mjesec, int godina) {

switch (mjesec)

{

case 4: case 6: case 9: case 11: return 30;

case 2:

return isLeap(godina) ? 29 : 28;

default:

return 31;

}

}

int getRandomValue(int from, int to) {

return rand() % (to - from + 1) + from;

}

struct Datum

{

int dan;

int mjesec;

int godina;

void ispis() {

cout << dan << "." << mjesec << "." << godina << endl;

}

bool isValid() {

if (dan < 1 || dan > 31 || mjesec < 1 || mjesec > 12 || godina < 1)

return false;

return dan <= getDaysForMonth(mjesec, godina);

}

int toInt() {

return 365 \* godina + 30 \* mjesec + dan;

}

};

// koristimo pokazivačke varijable za rad sa adresama objekata iz niza

// povratna vrijednost je adresa objekta koji ima vecu "datumsku" vrijednost

Datum\* max(Datum\* d1, Datum\* d2) {

return (d1->toInt() > d2->toInt()) ? d1 : d2;

}

Datum\* getNewestDate(Datum \* d, int size) {

Datum\* temp = d; // kreiranje pokazivača i setovanje istog na adresu prvog elementa

for (int i = 0; i < size; i++)

temp = max(temp, d + i);

return temp;

}

int main() {

srand(42);

int vel;

do

{

cout << "Unesite dimenzije niza datuma: " << endl;

cin >> vel;

} while (vel < 1);

// kreiranje dinamičkog niza datuma

Datum\* d = new Datum[vel];

cout << "Generisanje slucajnih datuma..." << endl;

for (int i = 0; i < vel; i++)

{

do

{

d[i].dan = getRandomValue(1, 31);

d[i].mjesec = getRandomValue(1, 12);

d[i].godina = getRandomValue(1, 2100); // postavit ćemo 2100 kao najveću moguću vrijednost

} while (!d[i].isValid()); // ponavljanje generisanja slučajnog datuma ukoliko je nevalidna vrijednost

}

for (int i = 0; i < vel; i++)

{

cout << "Datum " << i << ": ";

d[i].ispis();

}

// dohvaćanje adresa objekta sa najvećom "datumskom" vrijednošću atributa (dan,mjesec,godina)

Datum\* newest = getNewestDate(d, vel);

cout << "Najnoviji datum od unesenih je: ";

newest->ispis();

// dealokacija dinamičkog niza datuma

delete[] d;

d = nullptr;

return 0;

}

# 2. Rekurzije

1. Omogućiti unos prirodnog broja N (N > 0). Nakon toga generisati Hailstone-ove brojeve (sekvencu) za uneseno N i prebrojati koliko je koraka potrebno dok sekvenca ne dosegne vrijednost 1.

Napomena: Za rješenje zadatka obavezno koristiti rekurzivnu funkciju.

A white grid with black text

Description automatically generated

Input : N = 7

Output :

Hailstone Numbers: 7, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1

No. of steps Required: 17

**Objašnjenje:**

Za N = 7.

Brojevi će biti izračunati na sljedeći način:

7 // pocetna vrijednost

3 \* 7 + 1 = 22 // Posto je neparan broj

22 / 2 = 11 // Paran broj

3 \* 11 + 1 = 34 // Neparan broj

// .... i sve tako do 1

**Rješenje:**

#include <iostream>

using namespace std;

int c; //counter

int HailstoneNumbers(int N)

{

cout << N << " ";

if (N == 1 && c == 0) {

// N is initially 1.

return c;

}

else if (N == 1 && c != 0) {

// N is reduced to 1.

c++;

return c;

}

else if (N % 2 == 0) {

// If N is Even.

c++;

HailstoneNumbers(N / 2);

}

else if (N % 2 != 0) {

// N is Odd.

c++;

HailstoneNumbers(3 \* N + 1);

}

}

int main()

{

int N;

do

{

cout << "Enter the value of N: ";

cin >> N;

} while (N<1);

int x;

x = HailstoneNumbers(N);

cout << endl;

cout << "Number of Steps: " << x;

system("Pause>0");

return 0;

}

1. Napisati rekurzivnu funkciju koja pretvara broj unesen kao niz karaktera u tip podatka int.

Input: str = “1234”

Output: 1234

Input: str = “0145”

Output: 145

#include<iostream>

using namespace std;

// Recursive function to convert

// string to integer

int stringToInt(const char\* str)

{

// If the number represented as a string

// contains only a single digit

// then returns its value

if (strlen(str) == 1)

return (str[0] - '0');

// Recursive call for the sub-string

// starting at the second character

char\* temp = new char[strlen(str+1) + 1];

strcpy\_s(temp, strlen(str + 1) + 1, str + 1);

double y = stringToInt(temp);

// First digit of the number

double x = str[0] - '0';

// First digit multiplied by the

// appropriate power of 10 and then

// add the recursive result

// For example, xy = ((x \* 10) + y)

x = x \* pow(10, strlen(str) - 1) + y;

return int(x);

}

int main()

{

char temp[10];

cout << "Unesite broj (kao niz karaktera): ";

cin.getline(temp, 10);

cout << (stringToInt(temp)) << endl;

}

1. Pronaći aritmetičku sredinu svih elemenata niza koristeći rekurzivnu funkciju.

A screenshot of a computer

Description automatically generated

Input : 1 2 3 4 5

Output : 3

#include <iostream>

using namespace std;

float findMean(int arr[], int vel)

{

if (vel == 1)

return (float)arr[vel - 1];

else

return ((float)(findMean(arr, vel - 1) \* (vel - 1) +

arr[vel - 1]) / vel);

}

int main()

{

float Mean = 0;

const int vel = 5;

int arr[vel];

for (int i = 0; i < vel; i++)

{

cout << "arr[" << i << "]: ";

cin >> arr[i];

}

cout << "Result: " << findMean(arr, vel);

return 0;

}

1. Izračunati sumu cifara unesenog broja (broj mora biti pozitivan) koristeći rekurzivnu funkciju

Input : 45632

Output :20

#include <iostream>

using namespace std;

int sum\_of\_digits(int n)

{

if (n == 0)

return 0;

return (n % 10 + sum\_of\_digits(n / 10));

}

int main()

{

int num;

do

{

cout << "Enter the value of num: ";

cin >> num;

} while (num<1);

int result = sum\_of\_digits(num);

cout << "Sum of digits in " << num << " is " << result << endl;

return 0;

}

# 3. Zadaci za pripremu

Napisati program u kojem je potrebno unijeti odabrani broj znamenki počevši od znamenke najveće težinske vrijednosti, pa od njih sastaviti i ispisati prirodni broj. Unos znamenaka se prekida kada se unese broj manji od 0 ili veći od 9. Ispisati poruku ukoliko je dobiveni broj savršen. Ispis neka bude oblika:

*Primjer izlaza:*

Upisi znamenku:2

Upisi znamenku:8

Upisi znamenku:-1

...

Broj sastavljen od zadanih znamenaka je 28.

Broj je savršen.

Napisati program koji korisniku omogućava unos realnog broja x, cijelog broja m i odabir funkcije za proračun datog izraza. Korisnik bira funkciju unosom jednog od sljedećih karaktera: 's' (sin), 'c' (cos), 'q' (sqrt) i 'p' (pow). U slučaju da korisnik unese karakter koji ne odgovara niti jednoj od predefinisanih vrijednosti ispisati odgovarajuću poruku i ponoviti odabir. Funkciji pow postavite broj 2 za drugi parametar.

Za odabranu funkciju sin izraz je:

Broj faktora u izrazu odgovara unesenom broju m. U rješenju obavezno iskoristiti switch iskaz.

Poštujući sve faze procesa programiranje napisati program koji korisniku omogućava unos cijelog broja N, te izračunava geometrijsku sredinu svih neparnih brojeva u intervalu [1, N] koji nisu djeljivi sa brojem 5. Rezultat treba biti zaokružen na dvije decimale.

Geometrijska sredina n brojeva se računa po formuli:

Napisati program koji omogućava unos **pozitivnog** broja *n* te pronalazi njegovu najmanju cifru i formira novi broj u kojem je najmanja cifra zamijenjena cifrom '3'. Ukoliko uneseni broj n sadrži više istih cifara koje su najmanje, tada je potrebno zamijeniti svaku od njih. Dodatno program treba ispisati razliku unesenog i broja koji se dobije nakon zamjene najmanje cifre.

Napisati program koji će omogućiti korisniku unos dva minimalno trocifrena prirodna broja m i n (n>m, m>100, n<500). Zatim napraviti funkciju koja će vratiti aritmeticku sredinu svih srednjih cifara svih brojeva u rangu od m do n. Također ta funkcija treba ispisati najveću srednju cifru navedenog ranga.

U main funkciji kreirati niz karaktera veličine 200 (char *recenica*[200]) sa slijedećim inicijaliziranim vrijednostima „Most u Mostaru je Stari most a pored njega je jos jedan most“. Također, dat je niz karaktera veličine 4 (char *rijec*[4]) koji je inicijaliziran vrijednostima „most“. (*Nije potrebno praviti funkciju za unos. Možete odmah pri deklaraciji inicijalizirati nizove navedenim karakterima*) Potrebno je napraviti funkciju koja će primiti pomenute nizove kao argumente te pronaći i prebrojati koliko puta se ponavlja sekvenca karaktera iz niza *rijec* u nizu *recenica*. Voditi računa da se rječju smatra bilo koji niz karaktera odvojen space-om sa obje strane. Program u main funkciji treba ispisati broj pronađenih riječi.

Napisati program koji će omogućiti unos elemenata tipa integer u matricu dimenzija 10x10 te izvršiti transponiranje matrice (*zamjeniti redove i kolone kao na slici*). Zatim napraviti funkciju koja će izračunati aritmetičku sredinu svih parnih brojeva iznad glavne dijagonale i aritmetičku sredinu svih neparnih brojeva iznad sporedne dijagonale transponirane matrice. Provjeriti koja aritmetička sredina je veća i ispisati odgovarajuću poruku i dobivenu vrijednost u main programu.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

*Primjer transponirane matrice*

**Poštujući sve faze procesa programiranja**, napisati program koji rjesava slijedeći problem.

Date su firme A, B i C sa godišnjim prihodima:

A – prihod 10M Eur

B – prihod 7M Eur

C – prihod 5M Eur

sa slijedećim stabilnim fiksnim godišnjim rastom prihoda za koji pretpostavljamo da se neće mijenjati

A – 2.05%

B – 2.80%

C – 4.30%

Potrebno je pronaći koja firma će prva prestići firmu A u godišnjem prihodu i za koliko godina će se to desiti. Također potrebno je ispisati koliko će sve tri firme imati godišnji prihod u toj godini.

**(opt)**

Svaki paran broj može se prikazati kao suma dvaju prostih brojeva (tkzv. Goldbachovo pravilo). Napisati program koji će najprije učitati dva prirodna broja n1 i n2 (n2>n1). Zatim prikazati sve parne brojeve u intervalu n1 do n2 kao sumu dvaju prostih brojeva. U glavnom programu samo unijeti navedena dva prirodna broja i pozvati funkciju koja obavlja zadani posao.

Napomena: *Po želji kreirati dodatne funkcije da se olakša rješavanje zadatka (npr funkcija bool prostBroj(int))*

**(opt)**

Napisati program koji ostvaruje slijedece funkcionalnosti:

* Omoguciti unos dvije integer vrijednosti a i b uz uvjete a>200, b>a+100.
* U rangu a – b pronaci sve parne brojeve i ispisati ih u obliku zbira dva palindroma. (Palindrom je broj koji je jednak broju koji se dobije obrtanjem cifara tog broja. Npr. 14541 je palindrom jer mu je ista vrijednost i ako se napise u obrnutom poretku cifara, dok broj 123 nije palindrom jer u obrnutom poretku cifara se dobije broj 321 (123 != 321).

Primjer ispisa:

A picture containing text

Description automatically generated

**(opt)**

Kaprekar konstanta je broj koji se dobije na slijedeci nacin:

* Izabere se bilo koji cetverocifeni broj uz uvjet da sve cetiri cifre nisu iste.
* Zatim se cifre u broj sortiraju silazno, te uzlazno.
* Izracuna se razlika tako sto se oduzmu se ta dva sortirana broja (razlika=veci-manji)
* Proces se ponavlja sve dok se uzastopno ne pocne pojavljivati isti broj.

Taj broj koji se ponavlja naziva se Kaprekar konstanta. Napisati program koji prima bilo koji cetverocifreni broj kojem nisu sve cifre jednake a zatim:

* Omoguciti da program pronadje Kaprekar konstantu (zabranjeno je upisati broj u code) i
* Ispisati u koliko iteracija se dobila Kaprekar konstanta za uneseni broj

Pomoc:

do

{

…..

} while (razlika != prethodnaRazlika); // Na ovaj nacin cete detektovati ponavljanje. Ostatak zadatka mozete rijesiti standardno.

Primjer ispisa:

A picture containing text

Description automatically generated

Napisati program koji će učitati niz od 50 integer vrijednosti. Napisati funkciju koja će provjeriti da li se u nizu nalazi sekvenca od minimalno 4 uzastopna broja. Funkcija treba da provjeri i prebroji koliko takvih sekvenci postoji u nizu i da ispise broj takvih sekvenci.  
Npr. 256,985,**321,322,323,324**,456,457,458,503,**201,202,203,204**,658....

Napisati program koji će učitati elemente matrice dimenzija n x n (*const int n* vrijednost izaberite po želji) te ispisati je li matrica centralno simetrična s obzirom na središnji element. Ako jest, program treba ispisati 1, ako nije, ispisati 0, a ako je broj redova paran (pa nema središnjeg elementa), ispisati -1. Koristiti zasebne funkcije za **unos** elemenata i **provjeru** simetričnosti.  
Primjeri centralno-simetričnih matrica:

Napisati program koji simulira bacanje 3 kockice (jedna kockica ima 6 strana i na tim stranama su brojevi 1-6). Simuliranje bacanja svake kockice ostvariti funkcijom rand()%6+1.Simulirati konstanto bacanje sve tri kockice dok se u dva uzastopna bacanja ne desi da se dobiju isti brojevi na sve tri kockice (*npr. u šestom bacanju se dobiju brojevi 2,2,2 a u sedmom 4,4,4 na sve tri kockice*). Ispisati koliko je ukupno bilo bacanja dok se nije ispunio navedeni uslov. Nije potrebno tražiti bilo kakav unos od korisnika.

Napisati program koji će omogućiti unos cjelobrojnih parnih vrijednosti sa neparnim brojem cifara jednodimenzionalni niz od 20 elemenata. Zatim napraviti **funkciju** koja će u primati kao argument taj niz. Funkcija treba da izbaci iz niza sve brojeve djeljive sa 6 a da ostavi u nizu sve ostale brojeve. Izbacivanje brojeva se treba napraviti tako da se ostatak niza pomjeri u lijevo da ne bi bilo praznih elemenata. To pomjeranje je potrebno uraditi prilikom izbacivanja svakog elementa a ne na kraju posto su svi elementi djeljivi sa 6. **Obavezno voditi racuna o optimizaciji code-a koja posebno utiče na broj bodova u ovom zadatku.**

Neka je data matrica dimenzija 10x12 u kojoj svaki red predstavlja uposlenika firme, a svaka kolona mjesec u godini. Elementi matrice predstavljaju evidentirane plate po mjesecima i uposlenicima (plata je iznos sa dvije decimalne cifre iza zareza). Kreirati, a potom u glavnom programu testirati, sljedeće funkcije:

* Funkciju za unos plata za sve uposlenike za odabrani mjesec. Omogućiti poziv funkcije iz main funkcije više puta kako bi se unijeli podaci za sve mjesece u godini.
* Funkciju koja za odabranog uposlenika vrši ispis plata po mjesecima. Ukoliko je za neki od mjeseci, uposlanik bio na neplacenom otsustvu, njegova plata ce biti 0 i nece se ispisivati podatak za taj mjesec. Pored toga, funkcija kao rezultat vraća prosječnu platu za cijelu godinu za odabranog uposlenika ukljucujuci i mjesece otsustva.
* Funkciju koja pronalazi uposlenika sa najvecom prosjecnom platom i ispisuje iznos prosjecne plate zaokruzen na dvije decimale, te vraca index tog uposlenika.

Napisati program koji će omogućiti korisniku unos broja n (uz uslov unosa 10<=n<=1000). Zatim simulirati unos n slučajnih vrijednosti. Simuliranje unosa ostvariti funkcijom rand()%1000+1. Izračunati statističke podatke u kojem procentu se od n generisanih slucajnih vrijednosti pojavljuje prost broj.

Npr. Ako je generisano 10 brojeva(n=10): **5**,**7**,4,**13**,**17**,20,25,23,30,45, u ovom uzorku od 10 brojeva imamo 4 prosta broja pa je procenat prostih brojeva u ovom slucaju 40%.

Dat je 2D niz koji simulira šahovsku tablu. Omogućiti korisniku unos cjelobrojnih elemenata 2D niza tako da se u svako „crno“ polje unese parni broj sa neparnim brojem cifara a u „bijelo“ polje neparni broj sa parnim brijem cifara. Provjeriti da li je matrica simetricna po glavnoj dijagonali (*dakle da li je broj na poziciji 1.0 jedna broju na poziciji 0.1, na 0.2 jednak onome na 2.0, na 3.1 jednak onome na 1.3 itd.)* te ako jeste simetricna naci i ispisati index reda sa najvecim prosjekom elemenata, a ako nije u potpunosti simetricna naci i ispisati index kolone sa najmanjim prosjekom elemenata.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0.0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 |
| 1.0 | 1.1 | 1.2 | 1.3 | 1.4 | 1.5 | 1.6 | 1.7 |
| 2.0 | 2.1 | 2.2 | 2.3 | 2.4 | 2.5 | 2.6 | 2.7 |
| 3.0 | 3.1 | 3.2 | 3.3 | 3.4 | 3.5 | 3.6 | 3.7 |
| 4.0 | 4.1 | 4.2 | 4.3 | 4.4 | 4.5 | 4.6 | 4.7 |
| 5.0 | 5.1 | 5.2 | 5.3 | 5.4 | 5.5 | 5.6 | 5.7 |
| 6.0 | 6.1 | 6.2 | 6.3 | 6.4 | 6.5 | 6.6 | 6.7 |
| 7.0 | 7.1 | 7.2 | 7.3 | 7.4 | 7.5 | 7.6 | 7.7 |

Napisati program koji simulira bacanje 3 kockice (jedna kockica ima 6 strana i na tim stranama su brojevi 1-6). Simuliranje bacanja svake kockice ostvariti funkcijom rand()%6+1.Simulirati konstanto bacanje sve tri kockice dok se u dva uzastopna bacanja ne desi da se dobiju isti brojevi na sve tri kockice (*npr. u šestom bacanju se dobiju brojevi 2,2,2 a u sedmom 4,4,4 na sve tri kockice*). Ispisati koliko je ukupno bilo bacanja dok se nije ispunio navedeni uslov. Nije potrebno tražiti bilo kakav unos od korisnika

**Poštujući sve faze procesa programiranja** napisati program kojim će se izračunati 20 uzastopnih vrijednosti brzine i pređenog puta objekta u slobodnom padu u razmaku od po 3 sec. Prilikom racunanja potrebno je i ispisati na ekranu uzastopno brzinu i pređeni put zajedno za svaki period od 3sec. *Obratiti pažnju da se koriste ispravni tipovi podataka*.

Formula za brzinu slobodnog pada je: v=g\*t //g=9.81  
Formula za pređeni put pri slobodnom padu je: s=(g/2)\*pow(t,2) //g=9.81

Primjer izlaza:

A screenshot of a computer

Description automatically generated

Napisati funkciju koja kao parametare prima dva niza char recenica[200] i char rijec[10] odnosno rečenicu (niz od najviše 200 karaktera) i riječ (niz od najviše 10 karaktera) te u nizu recenica[200] pretrazuje i pronalazi sekvencu karaktera unesenih u niz rijec[10], te prebrojati koliko puta se ta sekvenca nalazi u recenici. Radi jednostavnosti rjesenja podrazumjeva se da i recenica i rijec koja se pretrazuje zavrsavaju tackom.

Npr. ako je data rečenica u nizu recenica[200]:

"Ja studiram na fakultetu jer zelim zavrsiti fakultet."  
i rijec[10]:  
„fakultet“, treba se ispisati poruka da je data rijec pronadjena 2 puta u recenici.

Napisati program koji će omogućiti korisniku da unese velicinu jednodimenzionalnog dinamickog niza integer vrijednosti. **Napisati rekurzivnu funkciju** koja će taj niz puniti serijom brojeva 2, 4, 8, 16, 32, 64, 128, 256, 512 itd., tako da je svaki slijedeci broj jednak prethodnom pomnozenom sa 2, pocevsi od broja 2. Rekurzija se prekida kad se popuni cijeli niz ili kad se desi overflow na integer varijabli. **U ovom zadatku je zabranjeno indexirati elemente niza uglastim zagradama. Obavezno koristiti aritmetiku pokazivaca**

Kreirati **dinamički** dvodimenzionalni niz čiji su elementi objekti tipa uposlenik (uposlenik je struktura čija su obilježja data u nastavku). Omogućiti korisniku da unese dimenzije niza.  
struct uposlenik  
{  
 char \*ID;  
 char \*imePrezime;  
 float \*plata;   
};

Redovi dvodimenzionalnog niza predstavljaju razlicite odjele firme u kojima rade razliciti radnici. (radi jednostavnosti rješenja podrazumjevati da svaki odjel ima isti broj radnika) Omogućiti korisniku unos svih podataka za sve radnike, te napraviti funkciju koja će pronaći najveći prosjek plate po odjelima firme te ispisati koji je to redni broj odjela (index reda sa najvecim prosjekom) i koji je najveći prosjek plate.

Kreirati dvodimenzionalni **dinamički** niz čiji su elementiobjekti tipa figura (figura je struktura čija su obilježja data u nastavku). Omogućiti korisniku da unese dimenzije niza te omogućiti unos elemenata.  
struct figura  
{  
 int ID;  
 char \*nazivFigure;  
 float \*visinaFigure;   
};  
Potrebno je napraviti funkciju koja će računati prosječnu visinu svih figura koje se nalaze na crnim poljima ispod sporedne dijagonale. Funkcija **treba vratiti pokazivac** na memorijsku lokaciju u kojoj je upisana ta vrijednost.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0.0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 |
| 1.0 | 1.1 | 1.2 | 1.3 | 1.4 | 1.5 | 1.6 | 1.7 |
| 2.0 | 2.1 | 2.2 | 2.3 | 2.4 | 2.5 | 2.6 | 2.7 |
| 3.0 | 3.1 | 3.2 | 3.3 | 3.4 | 3.5 | 3.6 | 3.7 |
| 4.0 | 4.1 | 4.2 | 4.3 | 4.4 | 4.5 | 4.6 | 4.7 |
| 5.0 | 5.1 | 5.2 | 5.3 | 5.4 | 5.5 | 5.6 | 5.7 |
| 6.0 | 6.1 | 6.2 | 6.3 | 6.4 | 6.5 | 6.6 | 6.7 |
| 7.0 | 7.1 | 7.2 | 7.3 | 7.4 | 7.5 | 7.6 | 7.7 |

Kreirati **dinamički** dvodimenzionalni niz čiji su elementi objekti tipa „vrijeme“ (vrijeme je struktura čija su obilježja data u nastavku). Omogućiti korisniku da unese dimenzije niza.  
struct vrijeme  
{  
 float \*temperatura;  
 float \*vlaznostZraka;  
 bool \*padavine;  
 char \*vrstaPadavine;   
 float \*kolicinaPadavine;   
};

Redovi dvodimenzionalnog niza predstavljaju dane a kolone sate u danu. Omogućiti korisniku unos svih podataka za sve sate i dane, te napraviti funkciju koja će pronaći ispisati dan sa najvisom i dan sa najnizom prosjecnom temperaturom. Potrebno je napraviti i funkciju koja ce racunati procenat sati sa temperaturom visom od 25.5 °C a u kojima je bilo padavina u odnosu na ukupan broj sati sa temperaturom visom od 25.5 °C. Takodjer je potrebno napraviti funkciju koja ce pronaci sat sa najvisom prosjecnom temperaturom u toku svih unesenih dana. **U ovom zadatku je zabranjeno indexirati elemente niza uglastim zagradama. Obavezno koristiti aritmetiku pokazivaca**

Neka je data matrica dimenzija 7x24 u kojoj svaki red predstavlja dan u sedmici, sat u danu. Matrica predstavlja tabelu za unos temperatura svaki sat u toku cijele sedmice. (Voditi racuna o tipovima podataka.) Kreirati, a potom u glavnom programu testirati, sljedeće funkcije:

* Funkciju za unos svih temperatura za odabrani dan. Omogućiti poziv funkcije iz main funkcije više puta kako bi se unijeli podaci za sve dane.
* Funkciju koja za odabrani dan vrši ispis temperatura po satima.
* Funkciju koja pronalazi i ispisuje dan sa najvišom i dan sa najnižom prosječnom temperaturom.
* Funkciju koja pronalazi sat u svim danima koji je imao najvišu temperaturu.

Funkciju koja prinalazi i ispisuje prosječnu temperaturu u podne (12h) svih dana

Omoguciti unos dva prirodna broja X i Y iz intervala od 10 do 5000. Kreirati program koji će pronaći i ispisati sve brojeve od X do Y (uključujući granične vrijednosti) za koje važi da je obrnuti broj djeljiv sa njegovom sumom cifara. *Za olaksanje rjesenja zadatka napraviti funkcije za obrtanje poretka cifara i za sumu cifara.  
Npr. Za broj 144 vrijedi da mu je suma cifara 9 i da je broj koji se dobije nakon što se njegove cifre obrnu, 441, djeljiv sa 9.*

Kreirati **dinamički** dvodimenzionalni niz čiji su elementi objekti tipa vozilo (vozilo je struktura čija su obilježja data u nastavku). Omogućiti korisniku da unese dimenzije niza.  
struct vozilo  
{  
 char \*markaVozila;  
 char \*brojSasije;  
 char \*tipVozila; //dozvoliti unos samo vrijednosti „A“, „B“ ili „C“  
 float \*potrosnja\_na\_sto\_km;   
};

Redovi dvodimenzionalnog niza predstavljaju razlicite firme u a kolone razlicita vozila u tim firmama. (radi jednostavnosti rješenja podrazumjevati da svaka firma ima isti broj vozila) Omogućiti korisniku unos svih podataka za sva vozila, te napraviti funkciju koja će pronaći i ispisati index firme najvećom potrošnjom goriva svojih vozila. Zatim pronaći i ispisati prosječnu potrošnju svih vozila tipa „C“ u cijeloj matrici. **U ovom zadatku je zabranjeno indexirati elemente niza uglastim zagradama. Obavezno koristiti aritmetiku pokazivaca**

Napisati program koji će omogućiti korisniku da unese velicinu jednodimenzionalnog dinamickog niza integer vrijednosti. **Napisati rekurzivnu funkciju** koja će taj niz puniti serijom brojeva 2, 4, 8, 16, 32, 64, 128, 256, 512 itd., tako da je svaki slijedeci broj jednak prethodnom pomnozenom sa 2, pocevsi od broja 2. Rekurzija se prekida kad se popuni cijeli niz ili kad se desi overflow na integer varijabli. Zatim omogućiti korisniku unos želejnog broja te novom rekurzivnom funkcijom pretražiti niz i ispisati poruku da li je uneseni broj pronađen i na kojem indexu u nizu. **U ovom zadatku je zabranjeno indexirati elemente niza uglastim zagradama. Obavezno koristiti aritmetiku pokazivaca**

Napisati program koji omogućava unos karaktera sve dok se ne popuni niz od 30 elemenata prema sljedećim pravilima:

* Elementi niza mogu biti samo velika slova i brojevi.
* Svaki naredni element u nizu treba da bude abecedno poslije prethodnog, odnosno u slučaju broja, dalje od nule na brojnoj osi (voditi računa da se međusobno porede samo slova i samo brojevi).

Izlaz iz programa treba da budu dobiveni elementi niza. Testni primjer je urađen za niz od 5 elemenata. (Pojasnjenje: *Ne radi se o kombinaciji tipova podataka char i int vec se i brojevi unose kao obicni karakter. Zadatak je dodatno pojednostavljen time sto znate da se pri unosu broja kao karaktera moze unijeti samo jedna cifra sto dodatno olaksava trazenje njegovog prethodnika. Nije potrebno znati napamet ASCII vrijednosti jer se poredjenje moze vrsiti i sa karakterom npr. x>’A’ ili x>’1’ sa apostrofima. Ukoliko ipak zelite porediti sa ASCII vrijednoscu imajte na umu da je ASCII vrijednost za karaktere ‘0’-‘9’ = 48-57 a za ‘A’-‘Z’ = 65-90*)

Unesite veliko slovo ili broj: D

Unesite veliko slovo ili broj: 3

Unesite veliko slovo ili broj: A

Unesite veliko slovo ili broj: F

Unesite veliko slovo ili broj: 0 (unos se ignorise)

Unesite veliko slovo ili broj: 4

Unesite veliko slovo ili broj: R

Elementi niza: D 3 F 4 R

**Poštujući sve faze procesa programiranja**, napisati program koji korisniku omogućava unos cijelog broja N, te izračunava geometrijsku sredinu svih neparnih brojeva u intervalu [1, N] koji nisu djeljivi sa brojem 5. Rezultat treba biti zaokružen na dvije decimale.

Geometrijska sredina n brojeva se računa po formuli:

Napisati funkciju koja kao parametar prima rečenicu (niz od najviše 200 karaktera) i iz nje uklanja svaku riječ koja počinje velikim slovom. Pored toga, funkcija treba da prebroji ukupan broj riječi u rečenici prije izmjene. Koristiti sljedeći prototip funkcije: int ukloniRijeci(char []).

Npr. ako je data rečenica:

"Stari most u Mostaru spaja obale rijeke Neretve."

funkcija je treba pretvoriti u:

"most u spaja obale rijeke"

Funkcija vraća broj riječi u polaznoj rečenici (prije odsijecanja ostalih riječi). Radi jednostavnosti rješenja, pretpostavite da je riječ bilo koji niz znakova razdvojen znakom razmak

Napisati program koji na osnovu unesenog cijelog broja N pronalazi i ispisuje srednju cifru ako broj N ima neparan broj cifara, odnosno aritmeticku sredinu (*tip float*) srednje dvije cifre ako broj N ima parni broj cifara, a zatim pronaci **najmanji** broj tipa int sa kojim se srednja cifra ili aritmeticka sredina srednjih cifri treba pomnoziti da bi se dobio broj koji je veci od inicijalnog zadanog broja.

Npr.

za broj 75631 srednja cifra je 6, **najmanja** integer vrijednost sa kojom treba pomnoziti broj 6 da bi se dobio broj veci od 75631 je **12606** jer je **12606**\*6=75636, **75636**>**75631**

ili

za broj 984354 aritmeticka sredina srednje dvije cifre je 3.5, **najmanja** integer vrijednost sa kojom treba pomnoziti broj 3.5 da bi se dobio broj veci od 984354 je **281245** jer je **281245**\*3.5=984357.5, **984357.5**>**984354**

Napisati funkciju koja kao parametare prima dva niza char recenica[200] i char rijec[10] odnosno rečenicu (niz od najviše 200 karaktera) i riječ (niz od najviše 10 karaktera) te u nizu recenica[200] pretrazuje i pronalazi sekvencu karaktera unesenih u niz rijec[10]. Radi jednostavnosti rjesenja podrazumjeva se da i recenica i rijec koja se pretrazuje zavrsavaju tackom.

Poslije pronalaska trazene rijeci, pomenutu rijec treba zamijeniti karakterima „FIT“ (*Pomoc: Treba voditi racuna koliko karaktera ima rijec koja je pronadjena i zamjenjena*)

Npr. ako je data rečenica u nizu recenica[200]:

"Ja studiram na fakultetu jer zelim biti expert."

i rijec[10]:

„fakultet“,

funkcija je treba pronaci rijec „fakultet“ u recenici i pretvoriti je u „FIT“:

" Ja studiram na FITu jer zelim biti expert." (***ima ukupno manje slova a zadrzan je samo po jedan space izmedju rijeci***)

Obavezno u main funkciji omoguciti unos recenice, zatim trazene rijeci i ispisati novu recenicu poslije zamjene.

Neka je dat 2D niz tipa int dimenzija 10x10. Kreirati, a potom u glavnom programu testirati, sljedeće funkcije:

1. Funkciju koja za vrsi generisanje podataka u 2D nizu na nacin da se u elementu sa indexom [0][0] upise vrijednost 1 a svaki slijedeci element da ima vrijednost prethodnog elementa pomnozenog sa 2 kao na slici. Generisanje se prekida ako se desi overflow na varijabli int.
2. Funkciju koja popunjenu matricu iz pod-zadatka a) transponuje tako sto joj zamjeni redove i kolone.
3. Funkciju koja kao parametar prima pomenuti niz i index kolone te **silazno** sortira kompletan 2D niz po izabranoj koloni na nacin da se kompletni redovi zamjene prema vrijednosti u izabranoj koloni (princip sortiranja po koloni kao u MS Excel tabeli)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1** | **2** | **4** | **8** | **16** | **32** | **64** | **128** | **256** | **512** |
| **1024** | **2048** | **itd.** | **itd.** | **itd.** | **itd.** | **itd.** | **itd.** |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

**Poštujući sve faze procesa programiranja** napraviti program koji pronalazi i ispisuje sve složene brojeve brojeve iz intervala m-n (10<m<100, 500<n<2000, m<n; ukoliko unesene vrijednosti nisu ispravne, učitavanje treba ponavljati), te pronalazi i ispisuje njihovu aritmetičku sredinu. Napraviti sljedeće funkcije:

* *bool slozeni* – koja će ispitivati da li je broj složeni i
* *ispis* – koja koja će ispisivati sve složene brojeve i vratiti main funkciji njihovu aritmetičku sredinu.

U glavnom programu je potrebno ispisati aritmetičku sredinu. *Složeni broj je svaki broj koji nije prosti, odnosno koji ima bar jednog djelioca osim broja 1 i samog sebe.*

Napisati program koji će uzastopno učitatvati cijele brojeve m i n (1≤m≤50, 1≤n≤500) i pronalaziti srednju cifru njihovog proizvoda/umnoška (srednju cifru ako proizvod/umnožak ima neparan broj cifara, odnosno aritmeticku sredinu (*tip float*) dvije srednje cifre ako proizvod/umnožak ima parni broj cifara). Program treba ispisivati srednju cifru samo ako je veća od prethodno izračunate srednje cifre. Prva pronađena srednja cifra se svakako ispisuje. Program se forsirano prekida ako korisnik unese n=0. Program se prirodno završava ako je pronađena srednja cifra 9.

Primjer izvršenja:

unesi m i n: 10, 15

srednja cifra/sredina proizvoda brojeva 10 i 15 (10\*15=150) je: 5

unesi m i n: 10, 42

srednja cifra/sredina proizvoda brojeva 10 i 42 (10\*42=420) je: ne ispisuje se

unesi m i n: 12, 31

srednja cifra/sredina proizvoda brojeva 12 i 31 (12\*31=372) je: 7

unesi m i n: 149, 20

srednja cifra/sredina proizvoda brojeva 149 i 20 (149 \*20=2980) je: 8.5

unesi m i n: 150, 0

forsirani prekid...

Neka je dat 2D niz tipa int dimenzija 5x5. Kreirati, a potom u glavnom programu testirati, sljedeće funkcije:

1. Funkciju koja vrši generisanje Fibonaccijevih brojeva u 2D nizu dok se niz ne popuni
2. Funkcija koja ispisuje najveće zajedničke djelitelje brojeva na suprotnim indeksima matrice koristeći Euklidov algoritam, kad se indeksi izjednače petlja se treba prekinuti:

*Euklidov algoritam(a,b):*

*Dok(b>0)*

*p=a;*

*a=b;*

*b = p mod b;*

*NZD=a;*

Gdje je 'p' pomoćna varijabla. Suprotni indeksi matrice su matrica[0,0] i matrica[5,5], zatim matrica[0,1] i matrica [5,4] itd .

Primjer:

Najveći zajednički djelitelj za brojeve 610 i 34 je 2. Najveći zajednički djelitelj za brojeve 8 i 2584 je 8 itd.

A grid of numbers with arrows

Description automatically generated

Omoguciti unos dva prirodna broja X i Y iz intervala od 10 do 5000 uz minimalni uslov raspona da je Y>X+100. Kreirati program koji će pronaći i ispisati svaki broj u rangu od X do Y (uključujući granične vrijednosti) za koje važi da suma njegovih cifara predstavlja prost broj.**Npr. Za broj 92 vrijedi da mu je suma cifara 11. Broj 11 je prost broj pa to znači da broj 92 ispunjava uslov.**

Napisati program koji će omogućiti korisniku da unese velicinu n jednodimenzionalnog dinamickog niza integer vrijednosti (n>2). **Napisati rekurzivnu funkciju** koja će taj niz puniti serijom brojeva 1, 3, 3, 9, 27..., tako da je svaki slijedeci broj jednak umnosku dva prethodna broja (*Prva dva elementa su 1 i 3.*) Rekurzija se prekida kad se popuni cijeli niz ili kad se desi overflow na integer varijabli. Zatim napraviti **drugu rekurzivnu funkciju** koja ce izracunati zbir svih elemenata tog niza (*Voditi racuna o tipu podatka zbog velicine broja*). **U ovom zadatku je zabranjeno indexirati elemente niza uglastim zagradama. Obavezno koristiti aritmetiku pokazivaca**

Kreirati dvodimenzionalni **dinamički** niz čiji su elementiobjekti tipa figura (figura je struktura čija su obilježja data u nastavku). Omogućiti korisniku da unese dimenzije niza te omogućiti unos elemenata. Osigurati da bude istri broj kolona i redova.  
Potrebno je napraviti funkciju koja će provjeriti da li su figure na bijelim poljima simetricno rasporedjene po visini u odnosu na glavnu dijagonalu(da li je visina fugure na poziciji 0,1 jednaka visini figure na poziciji 1,0 itd. za sva bijela polja). Funkcija **treba vratiti pokazivac** na memorijsku lokaciju u kojoj je upisan zbir visina figura ako je tabla simetricno rasporedjena, a broj -1 ako nije simetricno rasporedjena.

struct figura  
{  
 int ID;  
 char \*nazivFigure;  
 float \*visinaFigure;   
};

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0.0 | battle, checkmate, chess, figure, game, pawn icon0.1 | 0.2 | battle, bishop, checkmate, chess, figure, game icon0.3 | 0.4 | 0.5 | 0.6 | 0.7 |
| battle, checkmate, chess, figure, game, pawn icon1.0 | 1.1 | battle, checkmate, chess, figure, game, knight icon1.2 | 1.3 | 1.4 | 1.5 | 1.6 | 1.7 |
| 2.0 | battle, checkmate, chess, figure, game, knight icon2.1 | 2.2 | 2.3 | 2.4 | 2.5 | 2.6 | 2.7 |
| battle, bishop, checkmate, chess, figure, game icon3.0 | 3.1 | 3.2 | 3.3 | 3.4 | 3.5 | 3.6 | 3.7 |
| 4.0 | 4.1 | 4.2 | 4.3 | 4.4 | 4.5 | 4.6 | 4.7 |
| 5.0 | 5.1 | 5.2 | 5.3 | 5.4 | 5.5 | 5.6 | 5.7 |
| 6.0 | 6.1 | 6.2 | 6.3 | 6.4 | 6.5 | 6.6 | 6.7 |
| 7.0 | 7.1 | 7.2 | 7.3 | 7.4 | 7.5 | 7.6 | 7.7 |

Primjer nekoliko simetricnih pozicija

Kreirati dvodimenzionalni **dinamički** niz čiji su elementiobjekti tipa figura (figura je struktura čija su obilježja data u nastavku). Omogućiti korisniku da unese dimenzije niza te omogućiti unos elemenata. Osigurati da bude isti broj kolona i redova.  
Potrebno je napraviti funkciju koja će provjeriti da li su figure na bijelim poljima simetricno rasporedjene po visini u odnosu na crno polje sa druge strane table u odnosu na horizontalnu osu na sredini table kao na slici(da li je visina fugure na poziciji 0,1 jednaka visini figure na poziciji 7,1 itd. za sva bijela polja). Funkcija **treba vratiti pokazivac** na memorijsku lokaciju u kojoj je upisan zbir visina figura ako je tabla simetricno rasporedjena, a broj -1 ako nije simetricno rasporedjena.

struct figura  
{  
 int ID;  
 char \*nazivFigure;  
 float \*visinaFigure;   
};

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0.0 | battle, checkmate, chess, figure, game, pawn icon0.1 | 0.2 | battle, bishop, checkmate, chess, figure, game icon0.3 | 0.4 | 0.5 | 0.6 | 0.7 |
| 1.0 | 1.1 | battle, checkmate, chess, figure, game, knight icon1.2 | 1.3 | 1.4 | 1.5 | 1.6 | 1.7 |
| 2.0 | 2.1 | 2.2 | 2.3 | 2.4 | 2.5 | 2.6 | 2.7 |
| 3.0 | 3.1 | 3.2 | 3.3 | 3.4 | 3.5 | 3.6 | 3.7 |
| 4.0 | 4.1 | 4.2 | 4.3 | 4.4 | 4.5 | 4.6 | 4.7 |
| 5.0 | 5.1 | 5.2 | 5.3 | 5.4 | 5.5 | 5.6 | 5.7 |
| 6.0 | 6.1 | battle, checkmate, chess, figure, game, knight icon6.2 | 6.3 | 6.4 | 6.5 | 6.6 | 6.7 |
| 7.0 | battle, checkmate, chess, figure, game, pawn icon7.1 | 7.2 | battle, bishop, checkmate, chess, figure, game icon7.3 | 7.4 | 7.5 | 7.6 | 7.7 |

Primjer nekoliko simetricnih pozicija

Napisati program koji će omogućiti korisniku da unese velicinu jednodimenzionalnog dinamickog niza integer vrijednosti te ga inicijalizirati random brojevima u opsegu 1-100. Za generisanje random brojeva koristiti: **rand()%100+1**. Punjenje niza se prekida kad se popuni niz ili kad se tri puta uzastopno dobije isti random broj.  
**Napisati rekurzivnu funkciju** koja će racunati zbir svih elemenata tog niza. **U ovom zadatku je zabranjeno indexirati elemente niza uglastim zagradama. Obavezno koristiti aritmetiku pokazivaca**

Data je struktura:

struct Automobil

{

char \*Naziv; //niz karaktera

char \*Marka; //niz karaktera

char \*Tip; //niz karaktera

int \*godinaProizvodnje; //u rangu 2005-2022

float \*potrosnjaNaStoKm;

};

Kreirati dvodimenzionalni dinamički niz čiji su elementi objekti tipa Automobil. Omogućiti korisniku unos dimenzija niza. Kreirajte funkcije:

· za unos obilježja automobila – neka funkcija prima pokazivač na student. **Obavezno automatizirati unos tako da se u sva obilježja upisuju random vrijednosti. Možete koristiti priloženo rješenje za randomiziranje niza karaktera te int i float vrijednosti, ili randomiziranje napraviti sami.** **Zadatak se neće smatrati ispravnim ukoliko ovo nije urađeno.**,

· za ispis kompletnog 2D niza sa svim obilježjima svih automobila,

· Pretpostaviti da redovi predstavljaju firme a kolone automobile u firmi. Pronaći indeks firme sa najmanjom prosječnom potrošnjim automobila.

**U ovom zadatku je zabranjeno indexirati elemente niza uglastim zagradama. Obavezno koristiti aritmetiku pokazivaca**

Napisati program koji učitava prirodni broj n. Program treba ispisati najmanji prirodni broj m, veći ili jednak n, koji je potencija nekog prirodnog broja, tj. m = k^l, gdje su k i l ≥ 2 prirodni brojevi. **U ovom zadatku je zabranjeno koristiti pow funkciju za racunanje potencije. Potenciju računati korsteći rekurzivnu funkciju kojoj se proslijeđuje k i l i u kojoj je dozvoljena samo operacija množenja.***Za olaksanje rjesenja, mozete izabrati prototip funkcija po zelji*

Napisati program koji će omogućiti korisniku unos broja n (uz uvjet unosa 10<=n<=1000). Ispisati sve brojeve u rangu 0-n koji zadovoljavaju uvjete:

* da su prosti i

da su im cifre u rastucem redoslijedu (npr. broj 149 je prost broj kojem su cifre u rastucem redoslijedu

Date su strukture:

struct datum

{

int d, m, g;

};

struct student

{

int IB;

char ImeiPrezime[30];

datum rodjenja;

float prosjecnaOcjena;

};

Kreirati dvodimenzionalni dinamički niz čiji su elementi objekti tipa student. Kreirajte funkcije:

· za unos obilježja studenta – neka funkcija prima pokazivač na student;

· za ispis obilježja studenta – neka funkcija prima pokazivač na student;

· za validacija unesenog datuma rođenja – funkcija prima pokazivač na datum a vraća true ako

je datum validan (npr. 2.2.2002.) ili vraća false ako datum nije validan (npr. 17.23.100000.);

Pretpostaviti da redovi predstavljaju generacije a kolone predmete, pronaći indeks generacije sa najboljim prosjekom i indeks predmeta sa najlošijim prosjekom

Napisati program koji će omogućiti korisniku unos broja n (uz uslov 10 <= n <= 1000). Zatim simulirati unos n slučajnih vrijednosti. Simuliranje unosa ostvariti funkcijom rand() % 1000 + 1. Izračunati statističke podatke u kojem procentu se od n generisanih slučajnih vrijednosti pojavljuje savršen broj. *Za neki broj se kaže da je savršen ukoliko je jednak sumi svih svojih djelilaca (ne uključujući njega samog). Na primjer, 28 je savršen broj: njegovi djelioci su 1, 2, 4, 7 i 14, a 1 + 2 + 4 + 7 + 14 = 28. Jedini savršeni brojevi u opsegu od 1 do 100 su brojevi 6 i 28*

Kreirati dvodimenzionalni **dinamički** niz čiji su elementiobjekti tipa fudbaler (fudbaler je struktura čija su obilježja data u nastavku). Omogućiti korisniku da unese dimenzije niza te omogućiti unos elemenata. Redovi niza predstavljaju različite klubove, a kolone razlicite fudbalere. Radi jednostavnosti rješenja, pretpostaviti da svaki klub ima isti broj fudbalera. Ovisno o unesenoj poziciji (N, O ili V) identificira se da li je dati fudbaler napadac, odbrambeni ili vezni igrac. U razlicitim klubovima mogu igrati razliciti brojevi napadaca, veznih igraca ili odbrambenih igraca. (Npr. klub 1 moze imati 10 veznih, 3 napadaca i 5 odbrambenih igraca, dok klub 2 moze imati 12 veznih, 2 napadaca i 6 odbrambenih igraca i sl.)  
Izracunati i ispisati prosjecan broj golova po poziciji igraca za sve klubove. Zatim omoguciti korisniku izbor jednog tipa fudbalera (N, O ili V) i prema korisnickom izboru pronaci klub koja ima najbolji **prosjek** postignutih zgoditaka tog tipa igraca (npr. ako se unese slovo V za klub 1 racuna se prosjek za 10 veznih igraca dok se za klub 2 racuna prosjek 12 veznih igraca i ti prosjeci se porede).   
**U ovom zadatku je zabranjeno indexirati elemente niza uglastim zagradama. Obavezno koristiti aritmetiku pokazivaca.** **Obavezno** voditi računa o dealociranju dinamički alocirane memorije.

struct fudbaler

{

int ID;

char \*ime\_i\_prezime;

char \*pozcija; //prihvatiti samo slova N, O i V (napad, vezni i odbrana)

int \*broj\_golova;

};

Kreirati dvodimenzionalni **dinamički** niz čiji su elementiobjekti tipa uposlenik (uposlenik je struktura čija su obilježja data u nastavku). Omogućiti korisniku da unese dimenzije niza te omogućiti unos elemenata. Redovi niza predstavljaju različite IT firme, a kolone razlicite uposlenike. Radi jednostavnosti rješenja, pretpostaviti da svaka firma ima isti broj uposlenika. Ovisno o unesenom tipu (D, Q ili A) identificira se koji uposlenik obavlja koju duznost u firmi. U razlicitim firmama mogu biti razliciti brojevi developera, quality assurance ili architect-a. (Npr. firma 1 moze imati 10 developera, 3 quality assurance i 5 architect-a, dok druga firma moze imati 12 developera, 2 quality assurance i 6 architect-a i sl.)

struct uposlenik

{

int ID;

char \*ime\_i\_prezime;

char \*tip; //prihvatiti samo slova D, Q, A (developer, quality assurance i architect)

float \*plata;

};

Izracunati i ispisati prosjecne plate u firmama po tipu uposlenika. Zatim omoguciti korisniku izbor jednog tipa uposlenika (D, Q ili A) i prema korisnickom izboru pronaci firmu koja ima najbolju **prosjecnu** platu tog tipa uposlenika. (npr. ako se unese slovo D za firmu 1 racuna se prosjek za 10 developera dok se za firmu 2 racuna prosjek 12 developera i ti prosjeci se porede).

**U ovom zadatku je zabranjeno indexirati elemente niza uglastim zagradama. Obavezno koristiti aritmetiku pokazivaca.** **Obavezno** voditi računa o dealociranju dinamički alocirane memorije.

Dat je 2D niz dimenzija 8x8. Omogućiti korisniku unos cjelobrojnih elemenata 2D niza tako da se u svaki neparni red polje unesu parni brojevi a u svaki parni red neparni broj. Napraviti funkciju koja pronalazi kolonu sa najmanjim prosjekom elemenata. U istoj funkciji ispisati index pronađene kolone a u main funkciji sve elemente te kolone.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0.0 | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 |
| 1.0 | 1.1 | 1.2 | 1.3 | 1.4 | 1.5 | 1.6 | 1.7 |
| 2.0 | 2.1 | 2.2 | 2.3 | 2.4 | 2.5 | 2.6 | 2.7 |
| 3.0 | 3.1 | 3.2 | 3.3 | 3.4 | 3.5 | 3.6 | 3.7 |
| 4.0 | 4.1 | 4.2 | 4.3 | 4.4 | 4.5 | 4.6 | 4.7 |
| 5.0 | 5.1 | 5.2 | 5.3 | 5.4 | 5.5 | 5.6 | 5.7 |
| 6.0 | 6.1 | 6.2 | 6.3 | 6.4 | 6.5 | 6.6 | 6.7 |
| 7.0 | 7.1 | 7.2 | 7.3 | 7.4 | 7.5 | 7.6 | 7.7 |

Napisati funkciju koja kao parametar prima rečenicu (niz od najviše 200 karaktera) i iz nje uklanja svaku riječ koja ima više od 5 karaktera.  
Npr. ako je data rečenica: "U cetvrtak je bio lijep i suncan dan."  
funkcija je treba pretvoriti u: "U je bio lijep i dan "  
Radi jednostavnosti rješenja, pretpostavite da je riječ bilo koji niz znakova razdvojen znakom razmak

Napisati program koji omogućava korisniku unos graničnih vrijednosti ranga a,b (10<a<100 i 1000<b<2000).

Napisati funkciju koja će u unesenom rangu, uključujući i granične vrijendosti, pronaći i ispisati svaki broj za koji važi da je u isto vrijeme i prost i „palindrom“ odnosno da li je isti broj i kada se obrnu sve cifre. Po potrebi koristiti dodatne funkcije.

Npr. brojevi 383, 727, 757, 787, 797, 919, 929 su u isto vrijeme i prosti brojevi i palindromi.

Deklarirati kvadratnu matricu cijelih brojeva proizvoljnih dimenzija sa parnim brojem redova, te implementirati sljedeće funkcije:

* Funkciju za **unos** i formiranje elemenata matrice prema sljedećem obrascu: elemente neparnih redova matrice (1., 3., 5. itd.) čine minimalno trocifreni brojevi koje unosi korisnik programa (u slučaju unosa bilo kojeg broja koji ne zadovoljava postavljeni uslov, ponoviti unos); elementi parnih redova matrice se formiraju na osnovu elemenata iz prethodnog reda u istoj koloni matrice i to na način da je svaki novokreirani element matrice zapravo srednja cifra broja koji se u matrici nalazi iznad njega.
* Za pronalazak srednje cifre broja kreirati zasebnu funkciju **srednja\_cifra**, te je pozvati u funkciji unos. Srednju cifru brojeva sa parnim brojem cifara formirati kao srednju vrijednost dvaju cifara u sredini, npr. broj 4213 ima dvije srednje cifre, te funkcija treba da kao srednju cifru vrati vrijednost aritmetičke sredine cifri 2 i 1 zaokružene na veću decimalnu, u konkretnom slučaju broj 2.

Kreirati kratki testni program koji demonstrira funcionalnost kreiranih funkcija.

Primjer matrice dimenzija 4x4 i očekivanih rezultata je dat ispod.

|  |  |  |  |
| --- | --- | --- | --- |
| 2163 | 123 | 30452 | 671 |
| 4 | 2 | 4 | 7 |
| 7331 | 6113 | 741232 | 798 |
| 3 | 1 | 2 | 9 |