**UNIVERZITA SV. CYRILA A METODA V TRNAVE**

**FAKULTA PRÍRODNÝCH VIED**

**Zadanie 15.**

**Semestrálna práce**

**Programovanie**

|  |  |
| --- | --- |
| **2021** | **Ruslan Mursagulov, BC.** |

Obsah

[1. Zadanie 3](#_Toc92414584)

[2. Analýza 4](#_Toc92414585)

[3. Implementácia 5](#_Toc92414586)

[3.1. Knižnice 5](#_Toc92414590)

[3.2. # define 5](#_Toc92414591)

[3.3. Main 5](#_Toc92414592)

[3.4. Funkcia nasobenie 7](#_Toc92414593)

[3.5. Funkcia priemer\_stlpcev 9](#_Toc92414594)

[3.6. Funkcia all\_memory 10](#_Toc92414595)

[3.7. Funkcia dealokacia 12](#_Toc92414596)

[**4.** **Testovanie** 13](#_Toc92414597)

[4.1. Používanie prirodzených čísel pre riadky a stĺpce matice 13](#_Toc92414602)

[4.2. Používanie záporných čísel pre riadky a stĺpce matice 14](#_Toc92414603)

[4.3. Používanie desatinných čísel pre riadky a stĺpce matice 14](#_Toc92414604)

[5. Zhodnotenie 16](#_Toc92414605)

# Zadanie

Riešte problém, ktorý umožní používateľovi zadať prvky matíc A a B s rozmermi 2x3 a 3x3. Potom program vypočíta C, C=A\*B a nájde stĺpec matice C, ktorého priemer prvkov je najväčší. Prvky matíc sú reálne čísla a program vytvorte tak, aby rozmery matíc bolo možné jednoducho meniť. Na načítanie matíc a na výpočet priemeru zo stĺpca matice použite funkcie.

# Analýza

Podľa zadania môže používateľ nastaviť prvky matice A a B. Rozmery matíc A a B sú preddefinované, 2x3 a 3x3. Vieme, že na vytvorenie matíc musíme použiť dvojrozmerné dynamické polia. Najprv sa použije malloc, aby sa zabezpečilo, že používateľ môže nastaviť prvky matice. Aby sme skontrolovali, či sa matice vytvoria, overíme, či veľkosť matíc nebude nulová.  
  
Po tom, čo sme umožnili používateľovi zadať prvky pre matice A a B, musí program vynásobiť maticu A maticou B, aby sme mohli nájsť maticu C. Vieme, že na vynásobenie matíc sa počet stĺpcov prvej matice musí rovnať počtu riadkov druhej matice. Keďže počet riadkov a stĺpcov matíc je v našej prve úlohe na začiatku daný a je splnená vyššie uvedená podmienka, t. j. počet riadkov prvej matice sa rovná počtu stĺpcov druhej matice, môžeme nájsť tretiu maticu C.  
  
Ďalšou podmienkou zadania je, že program nájde stĺpec matice C, ktorého priemer prvkov je najväčší. V tejto časti program najprv zistí aritmetický priemer prvkov v každom stĺpci matice C a potom porovná, ktorý priemer je najväčší, a vypíše index pozície (stĺpca), v ktorej sa číslo nachádza.

# Implementácia



## Knižnice



Na volanie funkcií printf, scanf, malloc, free,exit boli použité knižnice **stdio.h** a **stdlib.h.**

## # define

Изображение выглядит как текст

Автоматически созданное описание

V zadaní bolo podmienkou, aby sa veľkosť matíc dala ľahko meniť. Na tento účel bol použitý direktíva preprocesora #define, aby nebolo potrebné zakaždým nastavovať veľkosť matice pre každú funkciu.

## Main

Изображение выглядит как текст

Автоматически созданное описание

Hlavná funkcia obsahuje hlavnú štruktúru kódu. Najprv sme deklarovali dátový typ matice pomocou **double**. Double sme použili preto, lebo úloha vyžaduje, aby sme použili reálne čísla. Ako vieme, reálne čísla sú čísla, ktoré obsahujú racionálne aj iracionálne čísla, a preto sme použili double, keďže float a double sa používajú na uchovávanie reálnych čísel.  
Okrem toho, aby mohol používateľ nastaviť parametre pre matice s veľkými hodnotami, pridelili sme pre tieto matice pamäť pomocou funkcie dealokacia. Táto funkcia bude podrobne opísaná v nasledujúcich častiach dokumentácie. Na zistenie pozícií (indexov) prvkov matice sme tiež použili dátový typ **integer.**  
Изображение выглядит как текст

Автоматически созданное описание  
  
Ďalej program pomocou cyklov **for** umožňuje používateľovi určiť prvky, t. j. riadky a stĺpce, pre matice A a B. Na tento účel sa používajú dva cykly **for.** Prvý, vonkajší cyklus sa používa na nájdenie riadkov matice. Druhý, vnútorný cyklus sa používa na nájdenie stĺpcov matice.

Изображение выглядит как текст

Автоматически созданное описание

V ďalšej časti kódu sa pomocou funkcie **printf** a cyklu **for** zobrazia riadky a stĺpce matíc A a B, ktoré nastavil používateľ.

  
  
  
  
  
Po zobrazení používateľom definovaných prvkov matice sa program prepne na funkciu nasobenie, aby našiel tretiu maticu, maticu C. Po vykonaní tejto funkcie sa program prepne na funkciu priemer\_stlpcev, ktorá zistí priemer každého stĺpca matice C a potom vypíše stĺpec, ktorý má najväčší priemer.

Изображение выглядит как текст

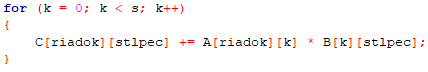
Автоматически созданное описание

Po vykonaní uvedených funkcií program, uvoľní pamäť, ktorá bola pridelená maticiam A,B a C, pomocou funkcie dealokacia. Táto funkcia tiež uvoľní pamäť alokovanú pre matice, aby nezaťažovala kód zbytočnou pamäťou.  
  
Ak sa program ukončí bez chýb po vykonaní všetkých funkcií, kód vráti nulu na identifikáciu úspešného vykonania programu.

## Funkcia nasobenie

Na úvode funkcie máme premenné dátového typu **int** riadok a stlpec, ktoré budú použité ako maticové indexy, to znamená, že budú označovať riadok a stĺpec konkrétneho prvku matice. Deklarovali sme aj premennú **int** k, aby sme ju mohli použiť pri násobení matíc A a B a hľadaní matice C.  
  
Изображение выглядит как текст

Автоматически созданное описание  
  
  
Predtým, ako popíšeme štruktúru kódu, chceme vysvetliť niektoré vlastnosti matice. Pri násobení matíc musí byť predovšetkým splnená podmienka, že počet stĺpcov prvej matice sa musí rovnať počtu riadkov druhej matice. Ako sme už uviedli, táto podmienka je splnená, pretože počet stĺpcov matice A sa rovná počtu riadkov matice B. Po overení tejto podmienky môžeme prejsť k násobeniu matíc a nájsť tretiu maticu.   
  
  
Na začiatku použijeme cyklus **for** na inicializáciu riadkov a stĺpcov matice C. Externý cyklus **for** sa používa na nájdenie riadkov a interný cyklus **for** sa používa na nájdenie stĺpcov. Pomocou týchto cyklov porovnáme počet **int** riadok a stĺpec s počtom premenné, ktoré boli vopred nastavené (**int** r\_1 a s).  
  
  
Pri každej iterácii cyklov inicializujeme všetky riadky a stĺpce matice C za predpokladu, že počet riadkov matice nie je väčší ako počet riadkov **int** r\_1 a počet stĺpcov nie je väčší ako počet stĺpcov **int** s. Po splnení týchto podmienok inicializujeme maticu C s nulovou hodnotou.   
  
  
Изображение выглядит как текст

Автоматически созданное описание  
  
  
Po inicializácii matice C môžeme prejsť k násobeniu matice. Na tento účel použijeme interný cyklus **for**. Ako vieme, na vynásobenie matíc je potrebné každý počet riadkov prvej matice vynásobiť každým počtom stĺpcov druhej matice.  
  
  
  
  
Pomocou cyklu **for** inicializujeme získané hodnoty riadkov a stĺpcov a pomocou funkcie **printf** vypíšeme výsledok, t. j. prvky matice C, na obrazovku.  
  
Изображение выглядит как текст

Автоматически созданное описание

## Funkcia priemer\_stlpcev

Na začiatku funkcie máme deklarované premenné dátového typu **int**(riadok a stlpec), premenné typu **double**(array\_priemer[stlpec], priemer = 0, sum = 0, max\_priemer) a premennú typu pointer na **double** \*average.   
Deklarujeme pointer na double \*average a potom ho priradíme do poľa **double** array\_priemer[stlpec], aby sme uložili hodnoty priemeru stĺpcov matice C

  
  
V ďalšej časti kódu použijeme cykly **for** na inicializáciu riadkov a stĺpcov matice a zistíme súčty čísel jednotlivých stĺpcov matice C. Súčet čísel stĺpcov sa zistí podľa vzorca sum += C[stlpec][riadok]. Po zistení súčtov stĺpcových čísel môžeme zistiť priemernú hodnotu prvkov každého stĺpca matice C. Na zistenie priemeru stĺpcov môžeme použiť vzorec priemer = suma / r\_1. Kde sum je súčet čísel stĺpcov a r\_1 je počet týchto čísel.   
Potom priradíme výsledok, t. j. priemeru hodnôt stĺpcov, do poľa (average\_riadok), aby sme ich hodnoty uložili ako prvky poľa a aby sme ich mohli neskôr porovnať. Na konci cyklu **for** sa pomocou funkcie printf zobrazí číslo každého stĺpca a jeho priemer.  
  
Изображение выглядит как текст

Автоматически созданное описание

V poslednej časti funkcie nájdeme stĺpec, ktorého prvky majú najvyššiu priemernú hodnotu. Najprv, aby sme našli najväčšiu hodnotu priemeru stĺpca matice C, priradíme indexu 0 (int max\_index) a samotnú hodnotu umiestnime na prvú pozíciu v poli (average[0]).   
 Ďalej použijeme cykly **for** a príkaz **if** na porovnanie priemeru stĺpcov matice. Na tento účel sa každá hodnota priemeru inicializovaná cyklom **for** porovná s prvým prvkom poľa pomocou príkazu **if**. Táto slučka bude pokračovať, kým sa neporovnajú všetky priemery. A po porovnaní sa priemer s najvyššou hodnotou priradí max\_priemer a jej pozícii sa priradí max\_index. Na konci funkcie sa zobrazí číslo stĺpca s najvyššou hodnotou priemeru.

Изображение выглядит как текст

Автоматически созданное описание

## Funkcia all\_memory

V tejto funkcii budeme alokovať pamäť pre dvojrozmerné pole, t. j. matice. Na úvode funkcie máme premenné dátového typu pointer na pointer double(matica), premenné typu int.  
  
Изображение выглядит как текст

Автоматически созданное описание   
  
Najprv pomocou funkcie malloc pridelíme pamäť pre maticu(pointer na pointer double). Zároveň môžeme skontrolovať, či sme pre dané pole pridelili dostatok pamäte. Na tento účel môžeme použiť príkaz **if** na kontrolu, či je pamäť rovná nule. Ak je táto podmienka splnená, môžeme v tele funkcie použiť príkaz printf na označenie, že nebolo alokované dostatočné pamäte. A potom funkcia exit ukončí proces   
  
  
Изображение выглядит как текст

Автоматически созданное описание  
  
  
V ďalšej časti kódu budeme alokovať pamäť pre každý riadok matice pomocou cyklu for. Príkaz if použijeme aj na kontrolu, či je alokovaný dostatok pamäte; ak je alokovaná pamäť rovná 0, použijeme funkciu printf na informovanie používateľa, že maticu nemožno vytvoriť. Tento proces sa končí funkciou exit. Po ukončení cyklu vráti funkcia premennú double(matica).   
  
Изображение выглядит как текст

Автоматически созданное описание

## Funkcia dealokacia

Keďže funkcia malloc nedokáže sama uvoľniť alokovanú pamäť, na jej uvoľnenie sa použije táto funkcia. Na začiatku funkcie máme deklarované premenné dátového typu int(i).  
  
  
  
  
Na uvoľnenie pamäte matice stačí uvoľniť pamäť riadkov matice. Nie je potrebné opisovať samostatný proces uvoľňovania pamäte v stĺpcoch matice, pretože stĺpce sú na riadkoch matice, a teda pri uvoľňovaní pamäte riadkov matice sa pamäť pre stĺpce uvoľňuje spoločne.  
Pomocou cyklu **for** inicializujeme každý riadok a uvoľníme alokovanú pamäť v tele cyklu pomocou funkcie **free.** Po skončení cyklu sa uvoľní aj pamäť matice pomocou funkcie **free** a veľkosť pamäte matice sa priradí nule.  
  
  
Изображение выглядит как текст

Автоматически созданное описание

# **Testovanie**



## Používanie prirodzených čísel pre riadky a stĺpce matice

Изображение выглядит как текст

Автоматически созданное описание

## Používanie záporných čísel pre riadky a stĺpce matice

Изображение выглядит как текст

Автоматически созданное описание

## Používanie desatinných čísel pre riadky a stĺpce matice

Изображение выглядит как текст

Автоматически созданное описание

# Zhodnotenie

Program spĺňa všetky podmienky úlohy čislo 15.  
  
  
  
**Efektívnosť**  
  
Program používa minimálny počet funkcií s cieľom získať riešenie v čo najkratšom čase.  
  
  
**Korektnosť**  
  
Program funguje bez chýb. Používateľ môže zadať akékoľvek prirodzené, záporné, desatinné alebo iné čísla.  
  
  
**Elementárnosť**  
  
Táto vlastnosť je tiež splnená, pretože boli použité algoritmy, ktoré pozostávajú z vykonateľných akcií.  
  
  
**Modifikovateľnosť**  
  
Program sa skladá z main a ďalších funkcií, ktoré sú volané v tele main. Každú funkciu možno zmeniť samostatne. Na zmenu veľkosti matíc sa použil aj predradený makroprocesor #define, v ktorom môže používateľ meniť hodnoty veľkosti, a preto nie je potrebné meniť každý riadok kódu, kde je daná hodnota uvedená.  
  
  
**User-friendly**

Program obsahuje jasné pokyny pre používateľa. Kod zobrazuje každú pozíciu čísla zadaného používateľom. Všetky správy, ktoré sa zobrazujú na obrazovke, sú jednoduché, jasné a presné.