**Задание**

Создать программу, которая умножает матрицу на число, транспонирует и складывает две матрицы, с использованием функции(можно с switch).

**Вот рекомендации из методички (там есть ошибки):**

Функции пользователя и динамическое распределение памяти

Цель работы: Выработка навыков разработки функций и компоновки программы из нескольких функций (файлов) на примерах решения типовых задач линейной алгебры (операций с массивами). Знакомство с методами обработки динамических массивов.

Основное задание

Составить текст программы, состоящей из функций, реализующих операции над матрицами произвольного размера, представленных в виде динамических массивов, в соответствии с вашим вариантом задания. Для этого необходимо:

Познакомиться с функциями input () и output (), предназначенных для ввода и вывода соответственно элементов матриц произвольного размера.

Разработать функции, осуществляющие набор матричных операций в соответствии с вашим вариантом.

Дополнить функцию main() необходимыми операторами и операндами.

Рекомендации по программированию

Рассмотрим пример функции, которая вводит с клавиатуры матрицу размером m x n :m – количество строк, n - количество столбцов, выделяет необходимую область памяти для размещения ее элементов и осуществляет ввод их значений в память ПЭВМ.

int \*\* input(int n, int m)

{

int i, j;

int \*\*a;

//Выделение динамической памяти для элементов матрицы

a= (int\*\*)malloc(n\*sizeof (int\*));

for(i=0; i<m; i++)

{

a[i]=(int\*)malloc(n\* sizeof(int\*));

for(j=0;j<n;j++)

{

a[i][j]=0; //Обнуление ячеек памяти

}

}

for(i=0; i<m; i++)

for(j=0;j<n; j++)

{

printf(“\n Введите элемент матрицы A(%d,%d) (элемент массива:)” ,i+1,j+1);

scanf(“%d”,&a[i][j]);

}

return a;

}

Для динамического выделения свободной памяти в данном фрагменте программы используется функция malloc(). Она возвращает указатель типа void. Для правильного использования значение этой функции надо преобразовать к указателю на соответствующий тип. При успешном выполнении операции malloc() возвращает указатель на первый байт свободной памяти требуемого размера. Если достаточного количества памяти нет, то возвращается значение 0 (нулевой указатель). Чтобы определить количество байт, необходимых для переменной, используют операцию sizeof.

Для освобождения динамической памяти используют функцию free с прототипом void \*free(void \*p); где \*p – указатель на первый байт выделенной памяти. Прототипы обеих функций находятся в заголовочном файле stdlib.h.

Динамическое распределение памяти удобно тогда, когда заранее неизвестно количество используемых переменных (в нашем случае - это m и n).

Для выделения динамической памяти можно так же использовать функцию

new тип [количество элементов].

Рассмотрим теперь функцию, осуществляющую вывод матрицы на экран дисплея.

void output (int \*\*z, int m, int n)

{

int i, j;

printf (“\n Результирующая матрица \n”);

for(i=0; i<m; i++)

{

for(j=0; j<m; j++)

printf(“%8d”, z[i][j] ) ;

printf(“\n”);

}

}

Для того, чтобы начать выполнение работы составьте главную функцию main(), в которую внесите описанные выше функции и функции обработки матриц в соответствии с вашим заданием. В начало программы нужно поместить директивы и декларации.

Примерный вид функции main().

#include<stdio.h>

#include<conio.h>

#include<alloc.h>

//Объявление шаблонов функцийобработки

int \*\*input (int, int);

int fproi (int\*\*, int\*\*, int, int);

……………………………

void ouput(int\*\*, int, int);

void main(void)

{

int m,n;

int \*\*p, \*\*q;

clrscr();

puts(“ Введите размер исходной матрицы”);

printf(“число строк=”);

scanf(“%d”,&m);

printf(“число столбцов=”);

scanf(“%d”,&n);

//Вызов функций

p= input (m,n);

output (p,m,n);

q= input (m,n);

output (p,m,n);

fproi(p,q,m,n);

……………..

}