|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** |
| Институт искусственного интеллекта |
| Кафедра Технологий Искусственного Интеллекта |

Практическая работа № 11

по дисциплине

«Процедурное программирование»

Обучающийся: Погосян С. А.

Группа: КВБО-07-23

Руководитель *Яковлев Д. А*

Москва 2023

**Тема: «Динамические массивы»**

**Цель лабораторной работы:**

Целью данной лабораторной работы освоить на практике работу с динамическими массивами.

**Описание работы:**

* Указатели. Типизированные указатели.
* Указатели и массивы.
* Адресная арифметика.
* Динамическое выделение памяти.
* Динамическое освобождение памяти

**Задание 1:**

1. Переписать пример программы, изменив ее таким образом, чтобы размеры массива и сами элементы генерировались с помощью датчика случайных чисел

**Код программы:**

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

int main() {

    srand(time(NULL));

    int \*\*ar;

    int i, j, n, \*mas;

    n = rand() % 10;

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

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

    for (i = 0; i < n; ++i) {

        mas[i] = rand() % 10;

        ar[i] = (int \*)malloc(mas[i] \* sizeof(int));

    }

    for (i = 0; i < n; ++i) {

        for (j = 0; j < mas[i]; ++j) {

            ar[i][j] = rand() % (10 + 10 + 1) - 10;

            printf("%5d ", ar[i][j]);

        }

        putchar('\n');

        free(ar[i]);

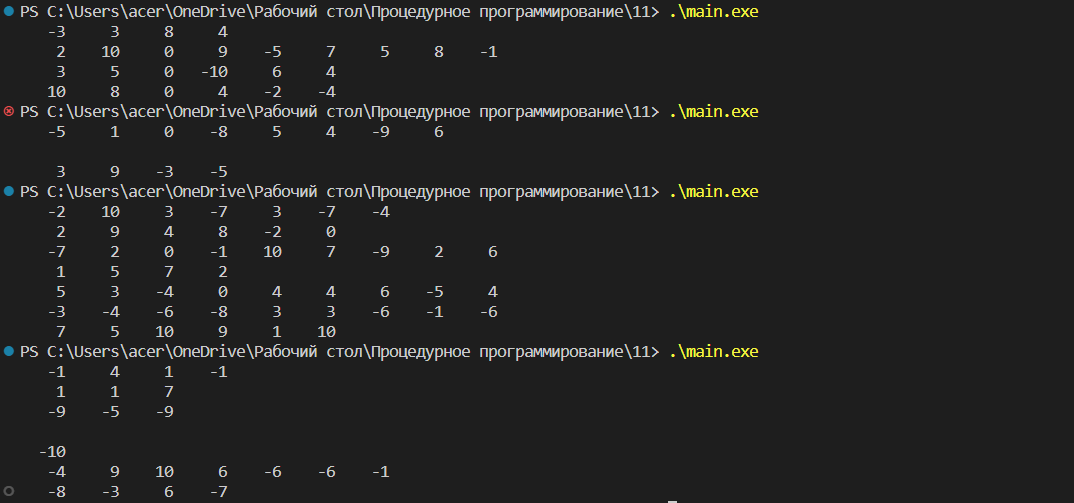
    }

    free(ar);

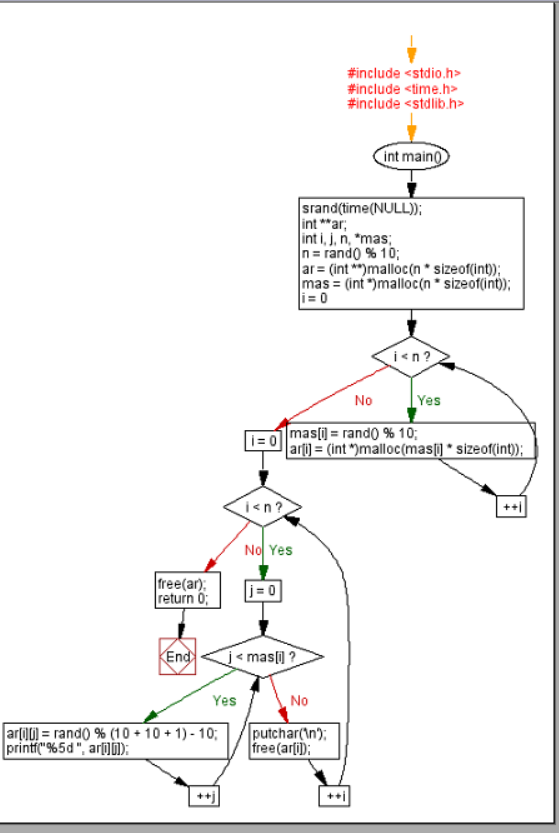
    return 0;

}

**Результата работы программы:**



**Алгоритм в виде блок-схемы:**

****

1. Выделить два массива A и B по 100 элементов типа int в динамической памяти с помощью функций calloc() и malloc() соответственно.
2. Вывести значения указателей A и B (адреса массивов в памяти) и содержимое массивов.

**Код программы:**

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

#define N1 100

#define N2 100

int main() {

    srand(time(NULL));

    int \*A, \*B;

    A = (int \*)malloc(N1 \* sizeof(int));

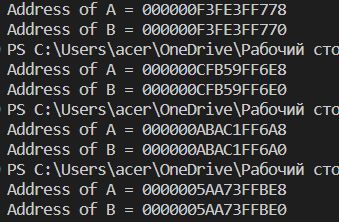
    B = (int \*)calloc(N2, sizeof(int));

    printf("Address of A = %p\n", &A);

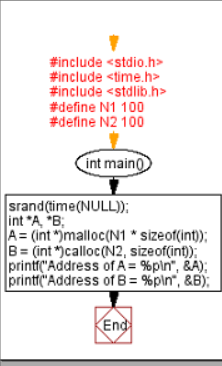
    printf("Address of B = %p\n", &B);

}

**Результат работы программы:**

****

**Алгоритм в виде блок-схемы:**

****

4. Перераспределить память:

1. увеличить размер A до 200 элементов;
2. размер B уменьшить до 50 элементов.
3. Посмотреть как изменились адреса A и B.
4. Изменить в пункте 3:
5. размер А уменьшить до 50 элементов.
6. увеличить размер В до 200 элементов.

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

#define N1 100

#define N2 100

int main() {

    srand(time(NULL));

    int \*A, \*B;

    A = (int \*)malloc(N1 \* sizeof(int));

    B = (int \*)calloc(N2, sizeof(int));

    printf("Initial Address of A = %p\n", &A);

    printf("Initial Address of B = %p\n", &B);

    A = (int \*)realloc(A, 200);

    B = (int \*)realloc(B, 50);

    printf("Result Address of A (200) = %p\n", &A);

    printf("Result Address of B (50) = %p\n", &B);

    A = (int \*)realloc(A, 50);

    B = (int \*)realloc(B, 200);

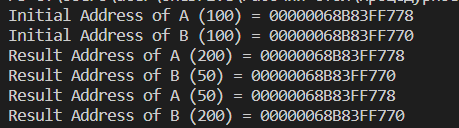
    printf("Result Address of A (50) = %p\n", &A);

    printf("Result Address of B (200) = %p\n", &B);

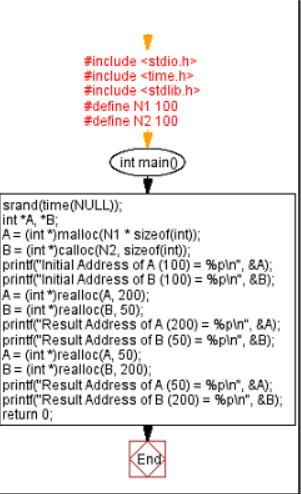
return 0;

}

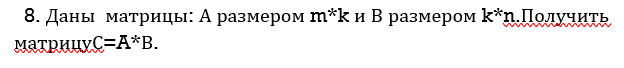
**Результат работы программы:**

****

**Алгоритм в виде блок-схемы:**

****

**Задание 2:**

****

**Код программы:**

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

#define M 5

#define K 3

#define N 6

int main() {

    srand(time(NULL));

    int \*\*A, \*\*B, \*\*C;

    int i, j, k;

    int cur\_total;

    A = (int \*\*)malloc(M \* sizeof(int));

    B = (int \*\*)malloc(K \* sizeof(int));

    C = (int \*\*)malloc(M \* sizeof(int));

    printf("A:\n");

    for (i = 0; i < M; ++i) {

        A[i] = (int \*)malloc(K \* sizeof(int));

        for (j = 0; j < K; ++j) {

            A[i][j] = rand() % 5;

            printf("%3d ", A[i][j]);

        }

        putchar('\n');

    }

    printf("\nB:\n");

    for (i = 0; i < K; ++i) {

        B[i] = (int \*)malloc(N \* sizeof(int));

        for (j = 0; j < N; ++j) {

            B[i][j] = rand() % 5;

            printf("%3d ", B[i][j]);

        }

        putchar('\n');

    }

    printf("\nC:\n");

    for (i = 0; i < M; ++i) {

        C[i] = (int \*)malloc(N \* sizeof(int));

        for (j = 0; j < N; ++j) {

           cur\_total = 0;

            for (k = 0; k < K; ++k) {

                cur\_total += A[i][k] \* B[k][j];

            }

            C[i][j] = cur\_total;

        }

    }

    for (i = 0; i < M; ++i) {

        for (j = 0; j < N; ++j) {

            printf("%3d ", C[i][j]);

        }

        putchar('\n');

    }

    free(A);

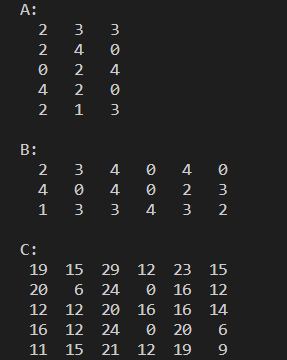
    free(B);

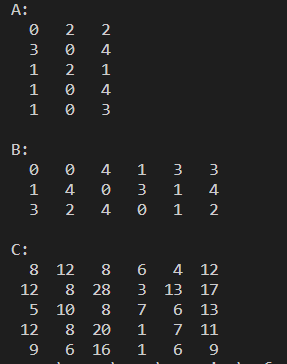
    free(C);

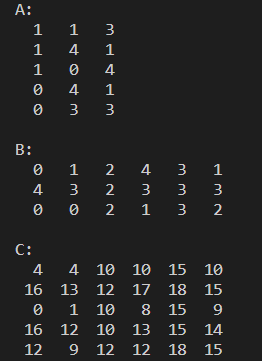
    return 0;

}

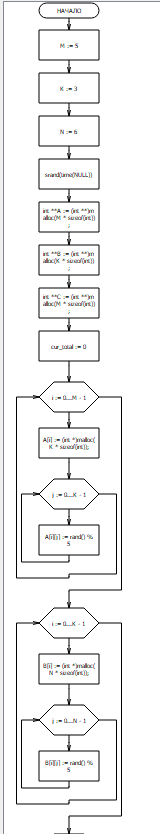
**Результат работы программы:**

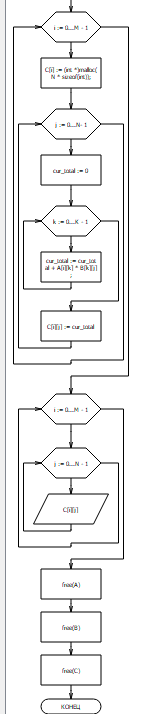
****





**Алгоритм в виде блок-схемы:**

****

****

**Контрольные вопросы:**

1. **Как организовать динамическое выделение памяти?**

Память для массива целых чисел в нашей работе не выделяется на этапе компиляции, так что нам достаточно объявить в программе только переменную - указатель на начало массива:

\*\*ar

-указатель на начало массива строк \* mas

Размерность массива определяется при выполнении программы, так что для ее сохранения нужна только одна переменная:

n

Элементы массива будут храниться в динамической памяти (куча), поэтому по окончании работы память занятую указателями необходимо освободить.

1. **Почему необходимо избегать утечки памяти?**

Утечка является проблемой, так как она блокирует ресурсы памяти, что со временем приводит к ухудшению производительности системы.

1. **В чем разница между функциями malloc() и calloc()?**

Функция malloc() выделяет область памяти заданного размера и возвращает указатель на эту область. Функция calloc() выделяет область памяти заданного размера и заполняет ее нулями.

1. **Смысл использования free()?**

Освобождение памяти из кучи

1. **Смысл работы функции realloc()?**

Функция realloc выполняет перераспределение блоков памяти.  
Размер блока памяти, на который ссылается параметр ptrmem изменяется на size байтов. Блок памяти может уменьшаться или увеличиваться в размере.

1. **Основные преимущества динамического выделения памяти?**

Контроль памяти**,** c помощью динамического выделения памяти под указатели строк можно размещать свободные массивы. Свободным называется двухмерный массив (матрица), размер строк которого может быть различным.