**Сопроводительный лист на издание в авторской редакции**

Название работы Динамические двумерные массивы в непрерывной памяти Методические указания к выполнению лабораторной работы № 24 для студентов очной формы обучения по направлениям подготовки 09.03.04 – «Программная инженерия», 09.03.01 «Информатика и вычислительная техника», 02.03.03 «Математическое обеспечение и администрирование информационных систем».– Брянск, 2017. – 7 с.

Актуальность и соответствующий   
научно-методический уровень подтверждаю \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись научного редактора)

Рукопись сверена и проверена автором \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись автора)

Рекомендуется к изданию \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись заведующего кафедрой)



**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**

**РОССИЙСКОЙ ФЕДЕРАЦИИ**

Брянский государственный технический университет

Утверждаю

Ректор университета

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О.Н. Федонин

«\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2017

ПРОГРАММИРОВАНИЕ

ДИНАМИЧЕСКИЕ ДВУМЕРНЫЕ МАССИВЫ В НЕПРЕРЫВНОЙ ПАМЯТИ

**Методические указания**

**к выполнению лабораторной работы № 24**

**для студентов очной формы обучения**

**по направлениям подготовки**

**09.03.04 – «Программная инженерия»,**

**09.03.01 «Информатика и вычислительная техника»,**

**02.03.03 «Математическое обеспечение и администрирование информационных систем»**

**Брянск 2017**

**УДК 004.**

Динамические двумерные массивы в непрерывной памяти [Текст] + [Электронный ресурс]: методические указания к выполнению лабораторной работы № 24 для студентов очной формы обучения по направлениям подготовки 09.03.04 – «Программная инженерия», 09.03.01 «Информатика и вычислительная техника», 02.03.03 «Математическое обеспечение и администрирование информационных систем».– Брянск, 2017. – 7 с.

Разработали:

Д. И. Булатицкий

канд. техн. наук, доц.,

Е..В. Коптенок

ассистент

Рекомендовано кафедрой «Информатика и программное обеспечение» БГТУ

**Методические издания публикуются в авторской редакции**

# Цель работы

Цель работы – научиться применять динамические двумерные массивы в непрерывной памяти для решения задач.

Продолжительность работы – 2 часа.

# ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

## Стандартные функции динамического выделения памяти

Функции динамического выделения памяти находят в оперативной памяти непрерывный участок требуемой длины и возвращают начальный адрес этого участка.

Функции динамического распределения памяти:

void\* malloc(РазмерМассиваВБайтах);  
void\* calloc(ЧислоЭлементов, РазмерЭлементаВБайтах);

Для использования функций динамического распределения памяти необходимо подключение библиотеки <malloc.h>:

#include <malloc.h>

Поскольку обе представленные функции в качестве возвращаемого значения имеют указатель на пустой тип void, требуется явное приведение типа возвращаемого значения.  
Для определения размера массива в байтах, используемого в качестве аргумента функции malloc() требуется количество элементов умножить на размер одного элемента. Поскольку элементами массива могут быть как данные простых типов, так и составных типов (например, структуры), для точного определения размера элемента в общем случае рекомендуется использование функции int sizeof(тип); которая определяет количество байт, занимаемое элементом указанного типа.

Память, динамически выделенная с использованием функций calloc(), malloc(), может быть освобождена с использованием функции free(указатель);

"Правилом хорошего тона" в программировании является освобождение динамически выделенной памяти в случае отсутствия ее дальнейшего использования. Однако если динамически выделенная память не освобождается явным образом, она будет освобождена по завершении выполнения программы.

## Динамическое выделение памяти для двумерных массивов в виде одномерного массива

Двумерная матрица будет располагаться в оперативной памяти в форме ленты, состоящей из элементов строк. При этом индекс любого элемента двумерной матрицы можно получить по формуле

**index = i\*m+j;**

где i - номер текущей строки; j - номер текущего столбца.

Объем памяти, требуемый для размещения двумерного массива, определится как n·m·(размер элемента)

Однако поскольку при таком объявлении компилятору явно не указывается количество элементов в строке и столбце двумерного массива, традиционное обращение к элементу путем указания индекса строки и индекса столбца является некорректным:  
a[i][j] - некорректно.

Правильное обращение к элементу с использованием указателя будет выглядеть как \*(p+i\*m+j),

где

* p - указатель на массив,
* m - количество столбцов,
* i - индекс строки,
* j - индекс столбца.

Пример.  Ввод и вывод значений динамического двумерного массива

#include <stdio.h>

#include <malloc.h>

#include <stdlib.h>

int main()

{

int \*a; // указатель на массив

int i, j, n, m;

printf("Введите количество строк: ");

scanf("%d", &n);

printf("Введите количество столбцов: ");

scanf("%d", &m);

// Выделение памяти

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

// Ввод элементов массива

for (i = 0; i<n; i++) // цикл по строкам

{

for (j = 0; j<m; j++) // цикл по столбцам

{

printf("a[%d][%d] = ", i, j);

scanf("%d", (a + i\*m + j));

}

}

// Вывод элементов массива

for (i = 0; i<n; i++) // цикл по строкам

{

for (j = 0; j<m; j++) // цикл по столбцам

{

printf("%5d ", \*(a + i\*m + j)); // 5 знакомест под элемент массива

}

printf("\n");

}

free(a);

getchar();

return 0;

}

## Создание динамического массива при помощи массива указаелей

Возможен также другой способ динамического выделения памяти под двумерный массив - с использованием массива указателей. Для этого необходимо:

* выделить блок оперативной памяти под массив указателей;
* выделить блоки оперативной памяти под одномерные массивы, представляющие собой строки искомой матрицы;
* записать адреса строк в массив указателей.

При таком способе выделения памяти компилятору явно указано количество строк и количество столбцов в массиве.

Пример

#include <stdio.h>

#include <malloc.h>

#include <stdlib.h>

int main()

{

int \*\*a; // указатель на указатель на строку элементов

int i, j, n, m;

printf("Введите количество строк: ");

scanf("%d", &n);

printf("Введите количество столбцов: ");

scanf("%d", &m);

// Выделение памяти под указатели на строки

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

// Ввод элементов массива

for (i = 0; i<n; i++) // цикл по строкам

{

// Выделение памяти под хранение строк

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

for (j = 0; j<m; j++) // цикл по столбцам

{

printf("a[%d][%d] = ", i, j);

scanf("%d", &a[i][j]);

}

}

// Вывод элементов массива

for (i = 0; i < n; i++) // цикл по строкам

{

for (j = 0; j < m; j++) // цикл по столбцам

{

printf("%5d ", a[i][j]); // 5 знакомест под элемент массива

}

printf("\n");

}

// Очистка памяти

for (i = 0; i < n; i++) // цикл по строкам

free(a[i]); // освобождение памяти под строку

free(a);

getchar();

return 0;

}

## Перераспределение памяти

Если размер выделяемой памяти нельзя задать заранее, например при вводе последовательности значений до определенной команды, то для увеличения размера массива при вводе следующего значения необходимо выполнить следующие действия:

* Выделить блок памяти размерности n+1 (на 1 больше текущего размера массива)
* Скопировать все значения, хранящиеся в массиве во вновь выделенную область памяти
* Освободить память, выделенную ранее для хранения массива
* Переместить указатель начала массива на начало вновь выделенной области памяти
* Дополнить массив последним введенным значением

Все перечисленные выше действия (кроме последнего) выполняет функция

void\* realloc (void\* ptr, size\_t size);

ptr - указатель на блок ранее выделенной памяти функциями malloc(), calloc() или realloc() для перемещения в новое место. Если этот параметр равен NULL, то выделяется новый блок, и функция возвращает на него указатель.

size - новый размер, в байтах, выделяемого блока памяти. Если size = 0, ранее выделенная память освобождается и функция возвращает нулевой указатель, ptr устанавливается в NULL.

Размер блока памяти, на который ссылается параметр ptr изменяется на size байтов. Блок памяти может уменьшаться или увеличиваться в размере. Содержимое блока памяти сохраняется даже если новый блок имеет меньший размер, чем старый. Но отбрасываются те данные, которые выходят за рамки нового блока. Если новый блок памяти больше старого, то содержимое вновь выделенной памяти будет неопределенным.

## «Свободные» массивы

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

Преимущество использования рваного массива заключается в том, что не требуется отводить память компьютера с запасом для размещения строки максимально возможной длины. Фактически рваный массив представляет собой одномерный массив указателей на одномерные массивы данных.

Для размещения в оперативной памяти матрицы со строками разной длины необходимо ввести дополнительный массив m, в котором будут храниться размеры строк (или осуществлять хранение размера в первом элементе массива).

# ХОД РАБОТЫ

Пример. Выделить память для ввода массива целых чисел. После ввода каждого значения задавать вопрос о вводе следующего значения.

#include <stdio.h>

#include <malloc.h>

int main()

{

int \*a = NULL, i = 0, elem;

char c;

do {

printf("a[%d]= ", i);

scanf("%d", &elem);

a = (int\*)realloc(a, (i + 1) \* sizeof(int));

a[i] = elem;

i++;

getchar();

printf("Next (y/n)? ");

c = getchar();

} while (c == 'y');

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

printf("%d ", a[j]);

getchar();

return 0;

}

В данной задаче при вводе каждого нового элемента массива происходит перераспределение памяти с помощью функции realloc().

**Пример2.** Создать динамический массив с различным количеством элементов в строках.

Для начала создадим массив указателей. Для каждого из элементов массива указателей выделим память в соответствии с количеством элементов в строке.

int main()

{

int \*\*a;

int i, j, n, \*m;

system("chcp 1251");

system("cls");

printf("Введите количество строк: ");

scanf("%d", &n);

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

m = (int\*)malloc(n \* sizeof(int)); // массив кол-ва элеменов в строках массива a

// Ввод элементов массива

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

{

printf("Введите количество столбцов строки %d: ", i);

scanf("%d", &m[i]);

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

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

printf("a[%d][%d]= ", i, j);

scanf("%d", &a[i][j]);

}

}

// Вывод элементов массива

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

{

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

{

printf("%3d ", a[i][j]);

}

printf("\n");

}

// Освобождение памяти

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

{

free(a[i]);

}

free(a);

free(m);

getchar(); getchar();

return 0;

}

# ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОГО ВЫПОЛНЕНИЯ

1. Сформировать в памяти динамический двумерный массив (двумя способами).
2. Заполнить массив значениями с клавиатуры.
3. Вывести массив на экран.
4. Выполнить согласно варианту:
5. Найти среднее арифметическое неотрицательных элементов матрицы, а также подсчитать, сколько таких элементов в каждой отдельно взятой строке матрицы.
6. Найти среднее арифметическое k-й строки и сумму элементов k-го столбца.
7. Найти сумму наибольшего и наименьшего элементов для каждой строки.
8. Найти сумму каждого столбца. Среди полученных сумм найти максимальное значение.
9. Найти сумму каждой строки. Среди полученных сумм найти максимальное значение.
10. В полученной матрице найти среднее арифметическое элементов столбца, в котором находится максимальный элемент матрицы В.
11. Найти число и сумму элементов матрицы , лежащих вне интервала (А;В).
12. Найти количество отрицательных чисел в каждой строке.
13. Найти максимальный среди минимальных значений по строке.

**Задания повышенной сложности:**

1. Записать в массив последовательно вводимые оценки студентов по предмету «Программирование». Ввод прекратить при вводе нуля. Определить средний балл в группе по предмету, количество неудовлетворительных оценок и количество отличников.
2. В течение полугода в каждый из месяцев производится определенное количество замеров температуры воздуха (количество замеров известно заранее, но в каждом месяце не известно). Определить, в каком из месяцев разница между максимальной и минимальной температурой самая большая.
3. В группе определенное количество студентов. Каждый из студентов имеет некоторое количество оценок по предмету. Количество оценок заранее не известно, но ввод оценок студента завершается вводом нуля. Определить средний балл для каждого из студентов, средний балл в группе, количество студентов, имеющих менее трех оценок и количество студентов, у которых только хорошие оценки (4 и 5).
4. Пусть дан двумерный массив из n строк, длина строк переменная и вводится пользователем вручную. Поменять местами строки с наибольшим и наименьшим элементом массива.

# КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какие функции для динамического выделения памяти существуют?
2. Какие способы создания динамического массива существуют?
3. В чем заключается перераспределение памяти?
4. Что такое «Свободные» массивы?

# Список рекомендуемой литературы

## Основная литература

1. Павловская Т.А. C#. Программирование на языке высокого уровня:учеб. Для вузов.-м. [и др.]:Питер,2014.-432
2. Давыдова Н.А.,боровская Е.В. Программирование: учеб. Пособие. М.:Бином. Лаб. Знаний,2011.-238 с.
3. Булатицкий Д.И. Объектно-ориентированное программирование: учеб. пособие.- Брянск: изд-во БГТУ,2008.-120 с.

## Дополнительная литература

1. Тюльпинова Н.В. Алгоритмизация и программирование: учеб. Пособие/Брян. Гос. Техн. Ун-т.- Брянск:изд-во БГТУ,2012.-199 с.-[+ электронная копия].
2. Иванова Г.С. Программирование: основы алгоритмизации и процедурное программирование, объектно ориентированное программирование: учеб. Для вузов.-2-е изд., стер.-м.:Кнорус, 2014.-425 с.

Динамические двумерные массивы в непрерывной памяти [Текст] + [Электронный ресурс]: методические указания к выполнению лабораторной работы № 24 для студентов очной формы обучения по направлениям подготовки 09.03.04 – «Программная инженерия», 09.03.01 «Информатика и вычислительная техника», 02.03.03 «Математическое обеспечение и администрирование информационных систем».– Брянск, 2017. – 7 с.

БУЛАТИЦКИЙ ДМИТРИЙ ИВАНОВИЧ

КОПТЕНОК ЕЛИЗАВЕТА ВИКТОРОВНА

Научный редактор В.Я. Израилев

Компьютерный набор Е.В. Коптенок

Иллюстрации Е.В. Коптенок

Подписано в печать 20.02.2017. Формат 60х84 1/15 Бумага офсетная. Офсетная печать. Усл.печ.л. \_\_\_ Уч.-изд.л. \_\_\_ Тираж 1 экз. Бесплатно

Брянский государственный технический университет

Кафедра «Информатика и программное обеспечение», тел. 56-09-84

241035, Брянск, бульвар 50 лет Октября, 7 БГТУ, тел. 58-82-49