Лабораторная работа 6.

«Указатели, арифметика указателей. Введение в функции.»

Комплект 6.

Задание 1.

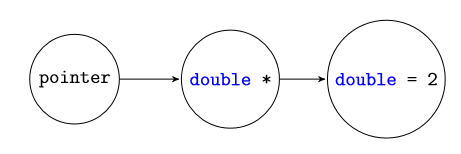
1. Внутри функции main(void) определите указатель double \*\*pointer = NULL;. В оперативной памяти создайте конструкцию, показанную на рисунке 1.

При этом выполните следующее:

• используйте функции типа \*malloc(...) для выделения оперативной памяти под динамические объекты;

• выведите число, указанное в крайней правой окружности, на экран, используя указатель double \*\*pointer = NULL;;

• используйте функцию free(...) для освобождения оперативной памяти, выделенную под динамические объекты.



|  |  |  |
| --- | --- | --- |
| Имя | Смысл | Тип |
| \*\*pointer | Указатель | double |



#include <stdio.h>

#include <stdlib.h>

int main() {

double \*\*pointer = NULL;

pointer = (double \*\*) malloc(sizeof(double \*));

pointer[0] = (double \*) malloc(sizeof(double));

\*pointer[0] = 2;

printf("pointer = %f", \*pointer[0]);

free(pointer);

return 0;

}





Задание 2.

1. Напишите программу, которая складывает два числа с использованием указателей на эти числа.

|  |  |  |
| --- | --- | --- |
| Имя | Смысл | Тип |
| a | Случайное число | int |
| b | Случайное число | int |
| \*pA | Указатель на число а | int |
| \*pB | Указатель на число b | int |



#include <stdio.h>

#include <stdlib.h>

int main() {

int a, b, \*pA, \*pB;

srand(time(NULL));

a = rand() % 101 - 50;

b = rand() % 101 - 50;

pA = &a;

pB = &b;

printf("a = %d, b = %d, a + b = %d", a, b, \*pA + \*pB);

return 0;

}



Задание 3.

1. Напишите программу, которая находить максимальное число из двух чисел, используя указатели на эти числа.

|  |  |  |
| --- | --- | --- |
| Имя | Смысл | Тип |
| a | Случайное число | int |
| b | Случайное число | int |
| \*pA | Указатель на число а | int |
| \*pB | Указатель на число b | int |



#include <stdio.h>

#include <stdlib.h>

int main() {

int a, b, \*pA, \*pB;

srand(time(NULL));

a = rand() % 101 - 50;

b = rand() % 101 - 50;

pA = &a;

pB = &b;

printf("a = %d, b = %d, max ", a, b);

if (\*pA > \*pB)

printf("%d", \*pA);

else

printf("%d", \*pB);

return 0;

}





Задание 4.

1. Напишите программу, которая создаёт одномерный динамический массив из чисел с плавающей точкой двойной точности, заполняет его значениями с клавиатуры и распечатывает все элементы этого массива, используя арифметику указателей (оператор +), а не обычный оператор доступа к элементу массива — [].

|  |  |  |
| --- | --- | --- |
| Имя | Смысл | Тип |
| n | Количество элементов массива | int |
| \*A | Массив | double |
| i | Параметр цикла | int |



#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

int main() {

char \*locale=setlocale(LC\_ALL, "Russian");

int n;

double \*A;

printf("Введите количество чисел: \n");

scanf("%d", &n);

A = (double \*)malloc(n \* sizeof(double));

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

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

scanf("%lf", (A + i));

}

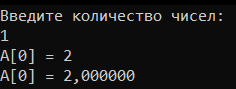
for (int i = 0; i < n; i++)

printf("A[%d] = %lf\n", i, \*(A + i));

free(A);

return 0;

}



Задание 5.

1. Вычислить факториал заданного числа, используя указатель на целое число, а просто не переменную целого типа.

|  |  |  |
| --- | --- | --- |
| Имя | Смысл | Тип |
| n | Число, вводимое с клавиатуры | int |
| a | Вспомогательная переменная | int |
| \*pA | Указатель | int |
| i | Параметр цикла | int |



#include <stdlib.h>

#include <stdio.h>

#include <locale.h>

int main() {

char \*locale=setlocale(LC\_ALL, "Russian");

int n, a = 1;

printf("Введите число: ");

scanf("%d", &n);

int \*pA = &a;

for (int i = 2; i <= n; i++)

\*pA \*= i;

printf("%d! = %d", n, \*pA);

return 0;

}





Задание 6.

1. Вывести элементы динамического массива целых чисел в обратном порядке, используя указатель и операцию декремента (--).

|  |  |  |
| --- | --- | --- |
| Имя | Смысл | Тип |
| \*A | Динамический массив | int |
| n | Количество элементов массива | int |
| i | Параметр цикла | int |



#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

int main() {

srand(time(NULL));

char \*locale=setlocale(LC\_ALL, "Russian");

int \*A, n;

printf("Введите количество элементов массива: ");

scanf("%d", &n);

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

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

A[i] = rand() % 101 - 50;

printf("A[%d] = %d\n", i, A[i]);

}

printf("Этот массив в обратном порядке: \n");

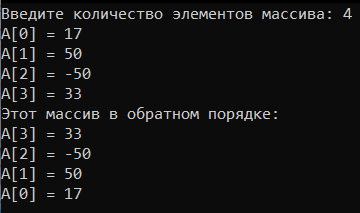
for (int i = n-1; i >= 0; i--)

printf("A[%d] = %d\n", i, \*(A+i));

free(A);

return 0;

}



Задание 7.

1. Отсортируйте заданный массив целых чисел, используя указатели, а не доступ по индексу ([]).

|  |  |  |
| --- | --- | --- |
| Имя | Смысл | Тип |
| n | Количество элементов массива | int |
| \*a | Динамический массив | int |
| i | Параметр цикла | int |
| j | Параметр цикла | int |
| buf | Буфер | int |



#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

int main() {

srand(time(NULL));

char \*locale=setlocale(LC\_ALL, "Russian");

int n, \*a;

printf("Количество элементов массива: ");

scanf("%d", &n);

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

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

a[i] = rand() % 101 - 50;

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

}

for (int i = 0; i < n - 1; i++) {

for (int j = n - 1; j > i; j--) {

if (\*(a + j - 1) > \*(a + j)) {

int buf = \*(a + j);

\*(a + j) = \*(a + j - 1);

\*(a + j - 1) = buf;

}

}

}

printf("Отсортированный массив:\n");

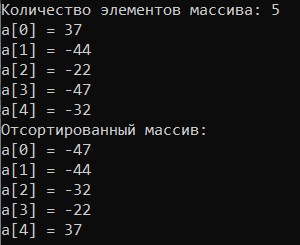
for (int i = 0; i < n; i++)

printf("a[%d] = %d\n", i, \*(a + i) );

free(a);

return 0;

}



Задание 8.

1. Определите переменную целого типа int a = 10; и выведите побайтово её содержимое на экран, используя указатель char \*.

|  |  |  |
| --- | --- | --- |
| Имя | Смысл | Тип |
| a | Целое число | int |
| \*b | Указатель | char |



#include <stdio.h>

#include <stdlib.h>

int main() {

int a = 10;

char \*b = (char\*) &a;

for (int i = 0; i < 4; i++)

printf("%d\n", \*(b + i));

return 0;

}



Задание 9.

1. Выделите память под двумерный динамический массив — матрицу — таким образом, чтобы данные все строки этой матрицы гарантированно располагались в оперативной памяти друг за другом (C 2D array contiguous memory allocation).

|  |  |  |
| --- | --- | --- |
| Имя | Смысл | Тип |
| m | Количество строк в матрице | int |
| n | Количество столбцов в матрице | int |
| \*\*A | Динамический массив, матрица | int |
| \*B | «2D массив» | int |
| i | Параметр цикла | int |
| j | Параметр цикла | int |



#include <stdio.h>

#include <stdlib.h>

int main() {

int m = 3, n = 2;

int \*\*A = (int\*\*)malloc(n \* sizeof(int\*) + n \* m \* sizeof(int));

int \*B = (int \*)((char \*)A + m \* sizeof(int \*));

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

A[i] = B + i \* n;

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

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

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

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

}

printf("----------\n");

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

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

printf("A[%d][%d] = %d\t", i, j,A[i][j]);

printf("\n");

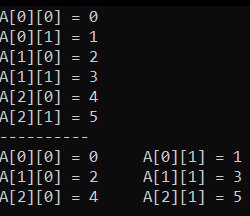
}

free(A);

return 0;

}





Комплект 7.

Задание 1.

1. Создайте две функции, которые вычисляют факториал числа, используя цикл:

• функцию, которая вычисляет факториал, используя цикл;

• функцию, которая вычисляет факториал, используя рекурсивный вызов самой себя.

Продемонстрируйте работу обеих функций.



|  |  |  |
| --- | --- | --- |
| Имя | Смысл | Тип |
| n | Целое число | int |
| r | Локальная пере | int |
| f | Переменная результата | int |
| i | Параметр цикла | int |
| fact\_c(int n) | Функция вычисления факториала через цикл | int |
| ract\_rec(int n) | Функция вычисления факториала через рекурсию | int |



#include <stdio.h>

#include <stdlib.h>

int fact\_c(int n) {

if (n == 1 || n == 0)

n = 1;

else

n = n \* fact\_c(n - 1);

return n;

}

int fact\_rec(int n) {

int r = 1;

for (int i = 1; i <= n; i ++)

r \*= i;

return r;

}

int main() {

int f = fact\_c(10);

printf("%d\n", f);

f = fact\_rec(10);

printf("%d", f);

return 0;

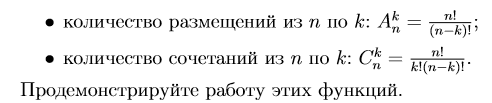
}





Задание 2.

1. Напишите отдельные функции для вычисления:





|  |  |  |
| --- | --- | --- |
| Имя | Смысл | Тип |
| n | Целое число, результат работы функции fact(n) | int |
| fact(int n) | Функция вычисления факториала числа n | int |
| Ank(int n, int k) | Количество размещений из n по k | int |
| Cnk(int n, int k) | Количество сочетаний из n по k | int |
| res | Переменная результата | int |



#include <stdio.h>

#include <stdlib.h>

int fact(int n) {

if (n == 1 || n == 0)

n = 1;

else

n = n \* fact(n - 1);

return n;

}

int Ank(int n, int k) {

return fact(n) / fact(n - k);

}

int Cnk(int n, int k) {

return fact(n) / (fact(k) \* fact(n - k));

}

int main() {

int res = Ank(5, 2);

printf("A(5, 2) = %d\n", res);

res = Cnk(5, 2);

printf("C(5, 2) = %d", res);

return 0;

}





Задание 3.

1. Объявите указатель на массив типа int и динамически выделите память для 12-ти элементов. Напишите функцию, которая поменяет значения четных и нечетных ячеек массива.

|  |  |  |
| --- | --- | --- |
| Имя | Смысл | Тип |
| swap(int\* a, int n) | Функция замены местами чётных и нечётных ячеек массива | int |
| i | Параметр цикла, индекс массива | int |
| n | Количество элементов массива | int |
| a[i] | Динамический массив, локальная переменная функции swap | int |
| \*array | Динамический массив | int |



#include <stdio.h>

#include <stdlib.h>

void swap(int\* a, int n) {

int i = 0;

while (i < n - 1) {

int t = a[i+1];

a[i+1] = a[i];

a[i] = t;

i += 2;

}

}

int main() {

int \*array;

int n;

printf("n = ");

scanf("%d", &n);

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

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

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

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

}

swap(array, n);

printf("---------\n");

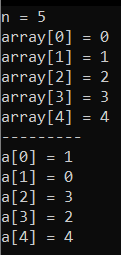
for (int i = 0; i < n; i++)

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

free(array);

return 0;

}



Задание 4.

1. Создать две основные функции:

• функцию для динамического выделения памяти под двумерный динамический массив типа double — матрицу;

• функцию для динамического освобождения памяти под двумерный динамический массив типа double — матрицу.

Создать две вспомогательные функции:

• функцию для заполнения матрицы типа double;

• функцию для распечатки этой матрицы на экране.

Продемонстрировать работу всех этих функций в своей программе.

|  |  |  |
| --- | --- | --- |
| Имя | Смысл | Тип |
| mcreate(m, n) | Функция выделения памяти под двумерный динамический массив | int\*\* |
| mfree(m, n, A) | Функция освобождения памяти от двумерного динамического массива | void |
| mfill(m, n, A) | Функция заполнения динамического массива случайными целыми числами | void |
| mprint(m, n, A) | Функция вывода двумерного массива | void |
| m | Количество строк в массиве | int |
| n | Количество столбцов в массиве | int |
| A | Динамический двумерный массив | int |
| i | Параметр цикла | int |
| j | Параметр цикла | int |



#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

int\*\* mcreate(int m, int n) {

int \*\*A;

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

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

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

return A;

}

void mfree(int m, int \*\*A) {

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

free(A[i]);

free(A);

}

void mfill(int m, int n, int \*\*A) {

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

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

A[i][j] = rand() % 101 - 50;

}

void mprint(int m, int n, int \*\*A) {

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

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

printf("A[%d][%d] = %d\t", i, j, A[i][j]);

printf("\n");

}

}

int main() {

srand(time(NULL));

char \*locale=setlocale(LC\_ALL, "Russian");

printf("Введите размер матрицы:\n");

int m, n;

printf("m = ");

scanf("%d", &m);

printf("n = ");

scanf("%d", &n);

int \*\*A = mcreate(m, n);

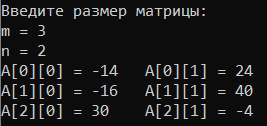
mfill(m, n, A);

mprint(m, n, A);

mfree(m, A);

return 0;

}



Задание 5.

1. Создать две функции для динамических массивов (для указателей):

• функцию умножения матрицы типа double на вектор-столбец типа double;

• функцию умножения одной матрицы типа double на другую матрицу типа double.

Продемонстрировать их работу на примере конкретных матриц и вектор-столбцов в своей программе, заполняя и распечатывая эти структуры с помощью отдельных функций, созданных ранее в задаче 7.4.

|  |  |  |
| --- | --- | --- |
| Имя | Смысл | Тип |
| i | Параметр цикла | int |
| j | Параметр цикла | int |
| r | Параметр цикла | int |
| matrix1 | Динамический двумерный массив 1, матрица 1 | double\*\* |
| n\_rows1 | Кол-во строк матрицы 1 | int |
| m\_columns1 | Кол-во столбцов матрицы 1 | int |
| vector | Вектор-столбец | double\* |
| n | Размер вектор-столбца | int |
| matrix2 | Матрица 2 | double\*\* |
| n\_rows2 | Кол-во строк матрицы 2 | int |
| m\_columns2 | Кол-во столбцов матрицы 2 | int |
| mfill(m, n, \*\*A) | Функция заполнения матрицы | void |
| mfree(m, \*\*A) | Функция освобождения памяти | void |
| mcreate(m, n) | Функция создания (выделения памяти) матрицы | double\*\* |
| mmultv(\*\*A, m, n, vector) | Функция умножения матрицы на вектор-столбец | void |
| mmult(l, q, k, \*\*A1, \*\*A2) | Функция умножения двух матриц | void |
| n\_rows | Аргумент функции (кол-во строк передаваемой матрицы) | int |
| m\_columns | Аргумент функции (кол-во столбцов передаваемой матрицы) | int |
| A | Аргумент функции (передаваемая матрица) | double\*\* |
| result | Локальная переменная функций multVector и multMatrix (служит как результат умножения) | double\* |
| l | Аргумент функции (размер передаваемой матрицы) | int |
| q | Аргумент функции (размер передаваемой матрицы) | int |
| k | Аргумент функции (размер передаваемой матрицы) | int |
| A1 | Аргумент функции (передаваемая матрица 1) | double\*\* |
| A2 | Аргумент функции (передаваемая матрица 2) | double\*\* |



#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

void mfill(int m, int n, double \*\*A) {

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

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

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

scanf("%lf", &A[i][j]);

}

}

void mfree(int m, double \*\*A) {

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

free(A[i]);

free(A);

}

double\*\* mcreate(int m, int n) {

double \*\*A;

A = (double \*\*)malloc(m \* sizeof(double \*));

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

A[i] = (double \*)malloc(n \* sizeof(double));

return A;

}

void mmultv(double \*\*A, int m, int n, double \*vector) {

double \*result;

result = (double \*) calloc(n, sizeof(double));

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

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

result[i] += vector[j] \* A[i][j];

printf("A[%d] = %lf\n", i, result[i]);

}

free(result);

}

void mmult(int l, int q, int k, double \*\*A1, double \*\*A2) {

double \*\*result;

result = mcreate(l, q);

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

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

result[i][j] = 0;

for (int r = 0; r < k; r++)

result[i][j] += A1[i][r] \* A2[r][j];

printf("A[%d][%d] = %lf\n", i, j, result[i][j]);

}

mfree(l, result);

}

int main() {

char \*locale=setlocale(LC\_ALL, "Russian");

double\*\* matrix1;

int n\_rows1, m\_columns1;

int choice;

printf("Enter size of 1st matrix:\nn = ");

scanf("%d", &n\_rows1);

printf("m = ");

scanf("%d", &m\_columns1);

matrix1 = mcreate(n\_rows1, m\_columns1);

mfill(n\_rows1, m\_columns1, matrix1);

double \*vector;

int n;

printf("Enter size of column-vector:\nn = ");

scanf("%d", &n);

if (n\_rows1 != n) {

printf("Error!");

return 1;

}

vector = (double \*) malloc(n \* sizeof(double));

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

printf("Vector[%d][0] = ", i);

scanf("%lf", &vector[i]);

}

mmultv(matrix1, n\_rows1, m\_columns1, vector);

free(vector);

double \*\*matrix2;

int n\_rows2, m\_columns2;

printf("Enter size of 2st matrix:\nn = ");

scanf("%d", &n\_rows2);

printf("m = ");

scanf("%d", &m\_columns2);

if (n\_rows1 != m\_columns2) {

printf("Error!");

return 1;

}

matrix2 = mcreate(n\_rows2, m\_columns2);

mfill(n\_rows2, m\_columns2, matrix2);

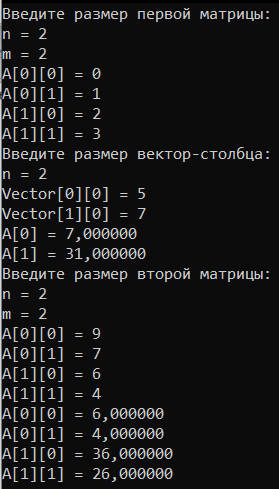
mmult(n\_rows1, m\_columns2, n\_rows2, matrix1, matrix2);

mfree(n\_rows2, matrix2);

mfree(n\_rows1, matrix1);

return 0;

}



Задание 6.

1. Создать функцию, которая вычисляет векторное произведение двух векторов в декартовых координатах, используя указатели на соответствующие массивы.

|  |  |  |
| --- | --- | --- |
| Имя | Смысл | Тип |
| A | Задаваемый вектор 1 | int\* |
| B | Задаваемые вектор 2 | int\* |
| result | Результативный вектор в главной функции (main) | int\* |
| i | Параметр цикла | int |
| n | Размер векторов | const |
| vfill() | Функция создания и заполнения вектора | int\* |
| vector | Возвращаемое значение полученного вектора в функции vfill | int\* |
| vmult(vector1, vector2) | Функция умножения двух векторов | int\* |
| vector1 | Передаваемый вектор 1 в функцию vmult | int\* |
| vector2 | Передаваемый вектор 2 в функцию vmult | int\* |
| res | Результат умножения векторов в функции vmult | int\* |



#include <stdio.h>

#include <stdlib.h>

#include <locale.h>

#define n 3

int\* vfill() {

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

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

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

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

}

return vector;

}

int\* vmult(int \*vector1, int \*vector2) {

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

\*(res + 0) = \*(vector1 + 1) \* \*(vector2 + 2) - \*(vector1 + 2) \* \*(vector2 + 1);

\*(res + 1) = \*(vector1 + 2) \* \*(vector2 + 0) - \*(vector1 + 0) \* \*(vector2 + 2);

\*(res + 2) = \*(vector1 + 0) \* \*(vector2 + 1) - \*(vector1 + 1) \* \*(vector2 + 0);

return res;

}

int main() {

char \*locale=setlocale(LC\_ALL, "Russian");

int \*A;

int \*B;

printf("Первый вектор A:\n");

A = vfill();

printf("Второй вектор B:\n");

B = vfill();

int \*result = vmult(A, B);

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

printf("AB[%d] = %d\n", i, result[i]);

free(A);

free(B);

free(result);

return 0;

}



