**Практическая работа № 3**

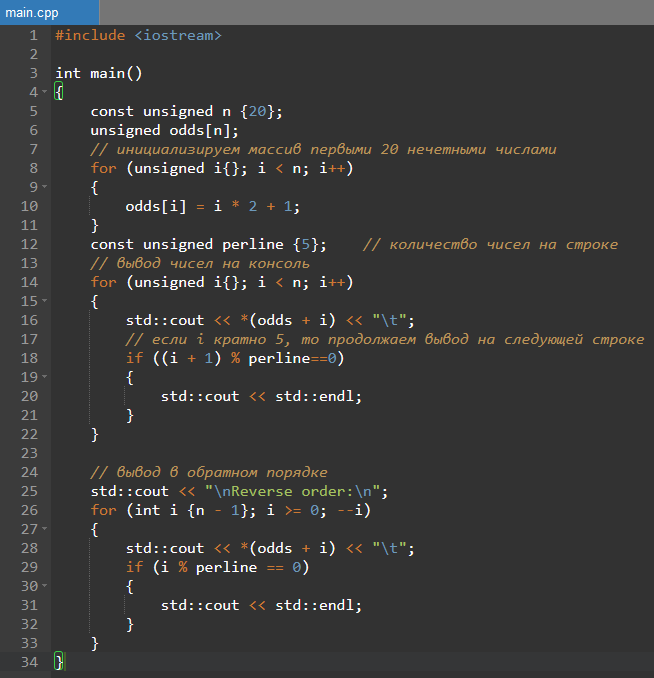
**Указатели в стиле C++. Динамическая память в C. Динамическая память в C++.**

**Цель работы:** изучение правил работы с указателями в C++.

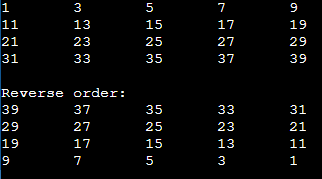
**Часть 1**

**Задача 1**. Напишите программу, которая определяет и инициализирует массив первыми 20 нечетными числами. Выведите числа из массива на консоль по пять в строку. Для вывода используйте нотацию указателей (имя массива в качестве указателя). Подобным образом с помощью указателя выведите элементы массива в обратном порядке.

Решение задачи 1.



Консоль:



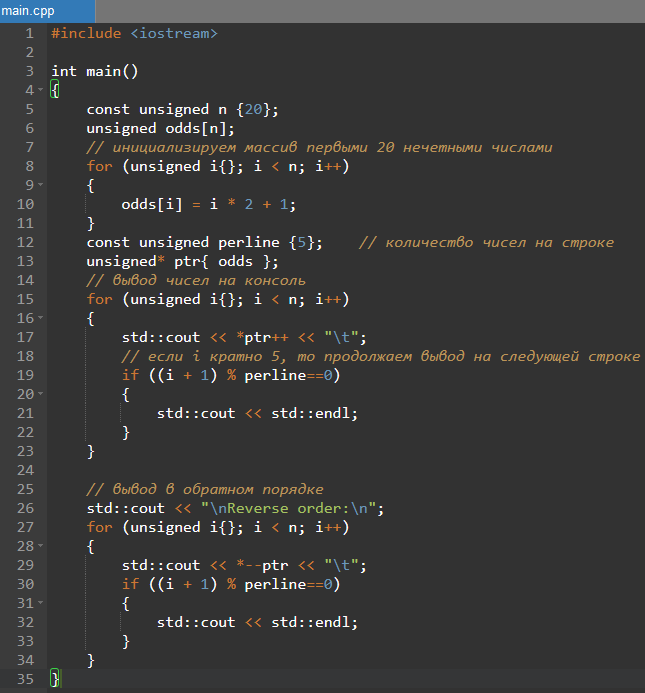
Программа откомпилирована в онлайн-компиляторе *https://www.onlinegdb.com/online\_c++\_compiler*

Таблица 1 – Варианты заданий для задачи № 1

|  |  |  |
| --- | --- | --- |
| № варианта | Количество элементов массива | Количество элементов, выводимых в строку |
| 1 | 24 | 6 |
| 2 | 25 | 5 |
| 3 | 28 | 7 |
| 4 | 30 | 6 |
| 5 | 35 | 7 |
| 6 | 36 | 6 |
| 7 | 40 | 5 |
| 8 | 42 | 6 |
| 9 | 24 | 4 |
| 10 | 28 | 4 |
| 11 | 30 | 5 |
| 12 | 35 | 5 |
| 13 | 40 | 8 |
| 14 | 42 | 7 |
| 15 | 24 | 4 |
| 16 | 44 | 11 |
| 17 | 44 | 11 |
| 18 | 48 | 4 |
| 19 | 48 | 12 |
| 20 | 52 | 13 |
| 21 | 56 | 7 |
| 22 | 56 | 14 |
| 23 | 60 | 15 |
| 24 | 64 | 8 |
| 25 | 72 | 8 |
| 26 | 72 | 9 |
| 27 | 81 | 9 |
| 28 | 100 | 10 |
| 29 | 36 | 12 |
| 30 | 28 | 14 |
| 31 | 32 | 4 |
| 32 | 32 | 8 |

**Задача 2.** Напишите программу, которая определяет и инициализирует массив первыми 20 нечетными числами. Выведите числа из массива на консоль по пять в строку. Для вывода определите указатель, который указывает на массив. С помощью инкремента (++) указателя выведите элементы в прямом порядке. А с потом в отдельном цикле с помощью декремента указателя выведите элементы массива в обратном порядке.

Решение задачи 2.



Консоль:

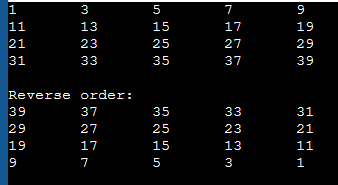


Таблица 2 – Варианты заданий для задачи № 2

|  |  |  |
| --- | --- | --- |
| № варианта | Количество элементов массива | Количество элементов, выводимых в строку |
| 1 | 35 | 7 |
| 2 | 36 | 6 |
| 3 | 40 | 5 |
| 4 | 42 | 6 |
| 5 | 24 | 6 |
| 6 | 25 | 5 |
| 7 | 28 | 7 |
| 8 | 30 | 6 |
| 9 | 35 | 5 |
| 10 | 40 | 8 |
| 11 | 42 | 7 |
| 12 | 24 | 4 |
| 13 | 44 | 11 |
| 14 | 44 | 11 |
| 15 | 48 | 4 |
| 16 | 48 | 12 |
| 17 | 52 | 13 |
| 18 | 56 | 7 |
| 19 | 56 | 14 |
| 20 | 60 | 15 |
| 21 | 64 | 8 |
| 22 | 72 | 8 |
| 23 | 72 | 9 |
| 24 | 81 | 9 |
| 25 | 100 | 10 |
| 26 | 36 | 12 |
| 27 | 28 | 14 |
| 28 | 32 | 4 |
| 29 | 32 | 8 |
| 30 | 24 | 6 |
| 31 | 25 | 5 |
| 32 | 28 | 7 |

**Задача № 3.** Решить с использованием указателей.

|  |  |
| --- | --- |
| № варианта | Задание |
| 1 | Найти сумму отрицательных элементов одномерного массива |
| 2 | Найти произведение положительных элементов одномерного массива |
| 3 | Найти среднее значение элементов одномерного массива |
| 4 | Найти среднее значение положительных элементов одномерного массива |
| 5 | Найти произведение модулей элементов одномерного массива |
| 6 | Найти произведение модулей отрицательных элементов одномерного массива |
| 7 | Найти минимальный элемент одномерного массива |
| 8 | Найти минимальный положительный элемент одномерного массива |
| 9 | Найти максимальный отрицательный элемент одномерного массива |
| 10 | Найти сумму положительных элементов одномерного массива |
| 11 | Найти произведение отрицательных элементов одномерного массива |
| 12 | Найти среднее значение отрицательных элементов одномерного массива |
| 13 | Найти сумму модулей элементов одномерного массива |
| 14 | Найти сумму модулей отрицательных элементов одномерного массива |
| 15 | Найти сумму модулей положительных элементов одномерного массива |
| 16 | Найти произведение модулей положительных элементов одномерного массива |
| 17 | Найти максимальный элемент одномерного массива |
| 18 | Найти минимальный нечетный элемент одномерного массива |
| 19 | Найти минимальный четный элемент одномерного массива |
| 20 | Найти максимальный элемент одномерного массива |
| 21 | Найти максимальный нечетный элемент одномерного массива |
| 22 | Найти максимальный четный элемент одномерного массива |
| 23 | Найти сумму отрицательных элементов одномерного массива |
| 24 | Найти произведение положительных элементов одномерного массива |
| 25 | Найти среднее значение элементов одномерного массива |
| 26 | Найти среднее значение положительных элементов одномерного массива |
| 27 | Найти произведение модулей элементов одномерного массива |
| 28 | Найти произведение модулей отрицательных элементов одномерного массива |
| 29 | Найти минимальный элемент одномерного массива |
| 30 | Найти минимальный положительный элемент одномерного массива |
| 31 | Найти максимальный отрицательный элемент одномерного массива |
| 32 | Найти сумму положительных элементов одномерного массива |

Часть 2.

**Динамическая память в C**

**Цель работы:** изучение правил работы с динамической памятью в стиле языка C.

**Выделение и освобождение памяти**

Для управления динамическим выделением памяти используется ряд функций, которые определены в заголовочном файле stdlib.h:

**malloc()**. Имеет прототип

*void \*malloc(unsigned s);*

Выделяет память длиной в *s* байт и возвращает указатель на начало выделенной памяти. В случае неудачного выполнения возвращает *NULL*.

**calloc()**. Имеет прототип

*void \*calloc(unsigned n, unsigned m);*

Выделяет память для n элементов по *m* байт каждый и возвращает указатель на начало выделенной памяти. В случае неудачного выполнения возвращает *NULL*.

**realloc()**. Имеет прототип

*void \*realloc(void \*bl, unsigned ns);*

Изменяет размер ранее выделенного блока памяти, на начало которого указывает указатель *bl*, до размера в *ns* байт. Если указатель *bl* имеет значение *NULL*, то есть память не выделялась, то действие функции аналогично действию *malloc*.

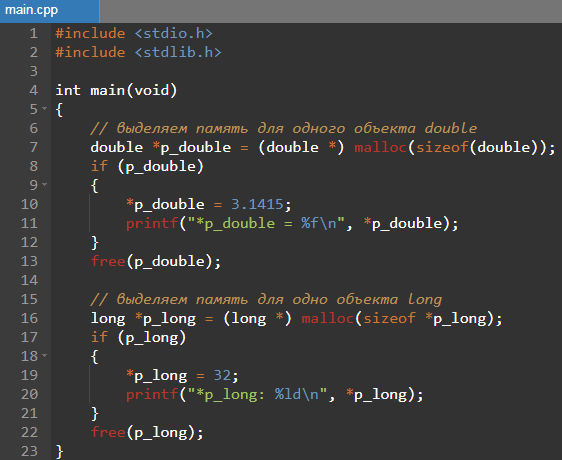
**free()**. Имеет прототип

*void \*free(void \*bl);*

Освобождает ранее выделенный блок памяти, на начало которого указывает указатель *bl*.

**Задача 4**. Напишите программу, которая выделяет память для одного объекта *double*, используя размер типа, и для одного объекта *long*, используя размер указателя. Поместите значения в выделенные участки памяти и выведите эти значения на консоль. После этого освободите выделенную память.

Решение задачи 1.



Следует обратить внимание на операторы приведения типа в строках 7 и 16:

***(double \*)*** и ***(long \*)***

Консоль:



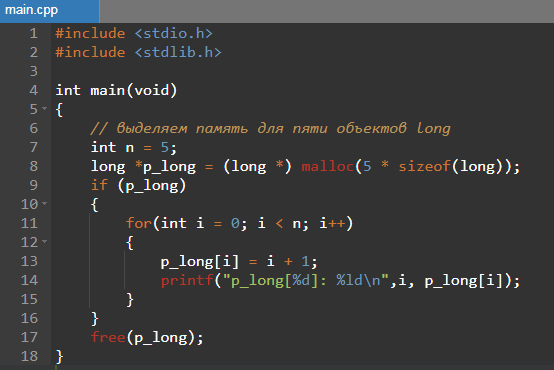
Программа откомпилирована в онлайн-компиляторе *https://www.onlinegdb.com/online\_c++\_compiler*

Таблица – Варианты заданий для задачи № 4

|  |  |  |
| --- | --- | --- |
| № варианта | Тип первой переменной | Тип второй переменной |
| 1 | float | unsigned long |
| 2 | long double | unsigned int |
| 3 | float | unsigned long long |
| 4 | long double | unsigned long |
| 5 | float | unsigned int |
| 6 | long double | unsigned long long |
| 7 | float | unsigned long |
| 8 | long double | int |
| 9 | float | long long |
| 10 | long double | unsigned long |
| 11 | float | unsigned int |
| 12 | long double | unsigned long long |
| 13 | float | unsigned long |
| 14 | long double | unsigned int |
| 15 | float | unsigned long long |
| 16 | long double | unsigned long |
| 17 | float | int |
| 18 | long double | long long |
| 19 | float | unsigned long |
| 20 | long double | unsigned int |
| 21 | float | unsigned long long |
| 22 | long double | unsigned long |
| 23 | float | unsigned int |
| 24 | long double | unsigned long long |
| 25 | float | unsigned long |
| 26 | long double | int |
| 27 | float | long long |
| 28 | long double | unsigned long |
| 29 | float | unsigned int |
| 30 | long double | unsigned long long |
| 31 | float | unsigned long |
| 32 | long double | unsigned int |

**Задача 5.** Напишите программу, которая динамически выделяет память для массива из пяти чисел типа *long*. Используя цикл *for*, присвойте значения элементами массива и выведите их на консоль. После этого освободите память.

Решение задачи 5.



Консоль:

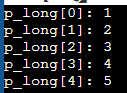
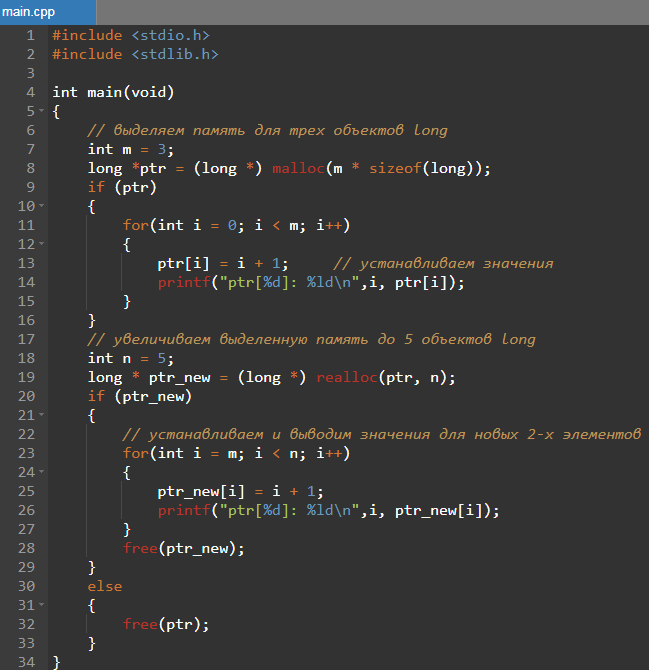


Таблица – Варианты заданий для задачи № 5

|  |  |
| --- | --- |
| № варианта | Доп. Задание |
| 1 | Найти сумму элементов массива, делящихся без остатка на 2 |
| 2 | Найти сумму элементов массива, делящихся без остатка на 3 |
| 3 | Найти сумму нечетных значений элементов массива |
| 4 | Найти произведение отрицательных элементов массива |
| 5 | Найти среднее значение положительных элементов массива |
| 6 | Найти среднее значение отрицательных элементов массива |
| 7 | Найти произведение четных значений элементов массива |
| 8 | Найти сумму квадратов элементов массива |
| 9 | Найти сумму четных значений элементов массива |
| 10 | Найти произведение положительных элементов массива |
| 11 | Найти произведение нечетных значений элементов массива |
| 12 | Найти сумму четных значений элементов массива |
| 13 | Найти сумму элементов массива, делящихся без остатка на 4 |
| 14 | Найти сумму элементов массива, делящихся без остатка на 5 |
| 15 | Найти произведение элементов массива, делящихся без остатка на 2 |
| 16 | Найти произведение элементов массива, делящихся без остатка на 3 |
| 17 | Найти сумму элементов массива, делящихся без остатка на 2 |
| 18 | Найти сумму элементов массива, делящихся без остатка на 3 |
| 19 | Найти сумму нечетных значений элементов массива |
| 20 | Найти произведение отрицательных элементов массива |
| 21 | Найти среднее значение положительных элементов массива |
| 22 | Найти среднее значение отрицательных элементов массива |
| 23 | Найти произведение четных значений элементов массива |
| 24 | Найти сумму квадратов элементов массива |
| 25 | Найти сумму четных значений элементов массива |
| 26 | Найти произведение положительных элементов массива |
| 27 | Найти произведение нечетных значений элементов массива |
| 28 | Найти сумму четных значений элементов массива |
| 29 | Найти сумму элементов массива, делящихся без остатка на 4 |
| 30 | Найти сумму элементов массива, делящихся без остатка на 5 |
| 31 | Найти произведение элементов массива, делящихся без остатка на 2 |
| 32 | Найти произведение элементов массива, делящихся без остатка на 3 |

**Задача 6.** Напишите программу, которая динамически выделяет память для массива из трех чисел типа *long*. Используя цикл *for*, присвойте значения элементами массива и выведите их на консоль. Затем с помощью функции *realloc*() увеличьте выделенную память до 5 элементов типа *long*. Установите значения для новых двух элементов *long* и выведите их на консоль. После этого освободите память.

Решение задачи 6.



Консоль:

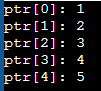
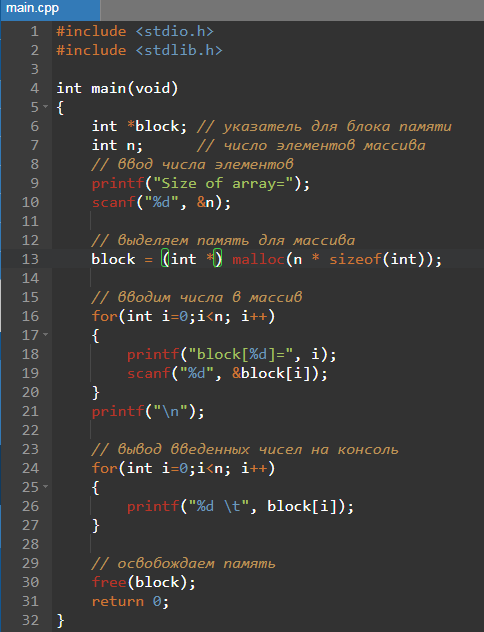


Таблица 6 – Варианты заданий для задачи № 6

|  |  |
| --- | --- |
| № варианта | Доп. задание |
| 1 | Добавить в конец массива один элемент, равный сумме элементов массива |
| 2 | Добавить в конец массива один элемент, равный произведению элементов массива |
| 3 | Добавить в конец массива один элемент, равный сумме положительных элементов массива |
| 4 | Добавить в конец массива один элемент, равный произведению отрицательных элементов массива |
| 5 | Добавить в конец массива два элемента, равные произведению и сумме элементов массива |
| 6 | Добавить в конец массива два элемента, равные произведению квадратов и сумме элементов массива |
| 7 | Добавить в конец массива два элемента, равные произведению и сумме квадратов элементов массива |
| 8 | Добавить в конец массива два элемента, равные среднеарифметическому положительных и отрицательных элементов массива |
| 9 | Добавить в конец массива один элемент, равный сумме элементов массива |
| 10 | Добавить в конец массива один элемент, равный произведению элементов массива |
| 11 | Добавить в конец массива один элемент, равный сумме положительных элементов массива |
| 12 | Добавить в конец массива один элемент, равный произведению отрицательных элементов массива |
| 13 | Добавить в конец массива два элемента, равные произведению и сумме элементов массива |
| 14 | Добавить в конец массива два элемента, равные произведению квадратов и сумме элементов массива |
| 15 | Добавить в конец массива два элемента, равные произведению и сумме квадратов элементов массива |
| 16 | Добавить в конец массива два элемента, равные среднеарифметическому положительных и отрицательных элементов массива |
| 17 | Добавить в конец массива один элемент, равный сумме элементов массива |
| 18 | Добавить в конец массива один элемент, равный произведению элементов массива |
| 19 | Добавить в конец массива один элемент, равный сумме положительных элементов массива |
| 20 | Добавить в конец массива один элемент, равный произведению отрицательных элементов массива |
| 21 | Добавить в конец массива два элемента, равные произведению и сумме элементов массива |
| 22 | Добавить в конец массива два элемента, равные произведению квадратов и сумме элементов массива |
| 23 | Добавить в конец массива два элемента, равные произведению и сумме квадратов элементов массива |
| 24 | Добавить в конец массива два элемента, равные среднеарифметическому положительных и отрицательных элементов массива |
| 25 | Добавить в конец массива один элемент, равный сумме элементов массива |
| 26 | Добавить в конец массива один элемент, равный произведению элементов массива |
| 27 | Добавить в конец массива один элемент, равный сумме положительных элементов массива |
| 28 | Добавить в конец массива один элемент, равный произведению отрицательных элементов массива |
| 29 | Добавить в конец массива два элемента, равные произведению и сумме элементов массива |
| 30 | Добавить в конец массива два элемента, равные произведению квадратов и сумме элементов массива |
| 31 | Добавить в конец массива два элемента, равные произведению и сумме квадратов элементов массива |
| 32 | Добавить в конец массива два элемента, равные среднеарифметическому положительных и отрицательных элементов массива |

**Задача 7.** Напишите программу, которая динамически выделяет память для массива. При этом длина массива неизвестна и вводится во время выполнения программы пользователем, и также значения всех элементов массива также вводятся пользователем.

Решение задачи 7.



Консоль:

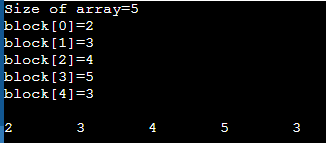


Таблица 7 – Варианты заданий для задачи № 7

|  |  |
| --- | --- |
| № варианта | Доп. задание |
| 1 | Отсортировать элементы массива по возрастанию |
| 2 | Отсортировать элементы массива по убыванию |
| 3 | Отсортировать элементы массива по возрастанию |
| 4 | Отсортировать элементы массива по убыванию |
| 5 | Отсортировать элементы массива по возрастанию |
| 6 | Отсортировать элементы массива по убыванию |
| 7 | Отсортировать элементы массива по возрастанию |
| 8 | Отсортировать элементы массива по убыванию |
| 9 | Отсортировать элементы массива по возрастанию |
| 10 | Отсортировать элементы массива по убыванию |
| 11 | Отсортировать элементы массива по возрастанию |
| 12 | Отсортировать элементы массива по убыванию |
| 13 | Отсортировать элементы массива по возрастанию |
| 14 | Отсортировать элементы массива по убыванию |
| 15 | Отсортировать элементы массива по возрастанию |
| 16 | Отсортировать элементы массива по убыванию |
| 17 | Отсортировать элементы массива по возрастанию |
| 18 | Отсортировать элементы массива по убыванию |
| 19 | Отсортировать элементы массива по возрастанию |
| 20 | Отсортировать элементы массива по убыванию |
| 21 | Отсортировать элементы массива по возрастанию |
| 22 | Отсортировать элементы массива по убыванию |
| 23 | Отсортировать элементы массива по возрастанию |
| 24 | Отсортировать элементы массива по убыванию |
| 25 | Отсортировать элементы массива по возрастанию |
| 26 | Отсортировать элементы массива по убыванию |
| 27 | Отсортировать элементы массива по возрастанию |
| 28 | Отсортировать элементы массива по убыванию |
| 29 | Отсортировать элементы массива по возрастанию |
| 30 | Отсортировать элементы массива по убыванию |
| 31 | Отсортировать элементы массива по возрастанию |
| 32 | Отсортировать элементы массива по убыванию |

Часть 3.

**Динамическая память в C++**

**Цель работы:** изучение правил работы с динамической памятью в стиле языка C++.

**1 Основные положения**

В C++ можно использовать различные типы объектов, которые различаются по использованию памяти. Так, **глобальные объекты** создаются при запуске программы и освобождаются при ее завершении. **Локальные автоматические объекты** создаются в блоке кода и удаляются, когда этот блок кода завершает работу. **Локальные статические объекты** создаются перед их первым использованием и освобождаются при завершении программы.

Глобальные, а также статические локальные объекты помещаются в статической памяти, а локальные автоматические объекты размещаются в стеке. Объекты в статической памяти и стеке создаются и удаляются компилятором. Статическая память очищается при завершении программы, а объекты из стека существуют, пока выполняется блок, в котором они определены. Когда блок завершает выполнение, то память в стеке, отведенная для переменных блока, освобождается. Стоит отметить, что память, выделяемая для стека, имеет ограниченный фиксированный размер.

В дополнение к этим типам в C++ можно создавать динамические объекты. Продолжительность их жизни не зависит от того, где они созданы. Динамические объекты существуют, пока не будут удалены явным образом. Динамические объекты размещаются в динамической памяти (free store). Это область памяти, не занятая операционной системой или другими загруженными в данный момент программами.

Использование динамических объектов имеет ряд преимуществ. Во-первых, более эффективное использование памяти - выделяется именно столько места, сколько необходимо, а после использования сразу освобождается. Во-вторых, мы можем использовать гораздо больший объем памяти, который в ином случае был бы не доступен. Но это и накладывает ограничения: мы должны следить, чтобы все динамические объекты были удалены.

Для управления динамическими объектами применяются операторы **new** и **delete**.

Оператор **new** выделяет место в динамической памяти для объекта и возвращает указатель на этот объект.

Оператор **delete** получает указатель на динамический объект и удаляет его из памяти.

**2 Выделение памяти**

Создание динамического объекта:

*int \*ptr{new int};*

*// или так*

*int \*ptr = new int;*

Оператор **new** создает новый объект типа int в динамической памяти и возвращает указатель на него. Таким образом, указатель ptr содержит адрес выделенной памяти. Значение такого объекта неопределено.

Также можно инициализировать объект при выделении памяти:

*int \*ptr{new int()}; // значение по умолчанию - 0*

*// int \*ptr = new int(); - или так*

*std::cout << \*ptr << std::endl; // 0*

Здесь объект в памяти, на который указывает указатель ptr, получает значение по умолчанию - число 0.

Для инициализации можно использовать фигурные скобки:

*int \*ptr{new int{}}; // значение по умолчанию - 0*

*// int \*ptr = new int{}; - или так*

*std::cout << \*ptr << std::endl; // 0*

Также можно инициализировать объект некоторым конкретным значением, например:

*int \*ptr{new int{5}};*

*// альтернативные варианты*

*// int \*ptr = new int{5};*

*// int \*ptr {new int(5)};*

*// int \*ptr = new int(5);*

*std::cout << \*ptr << std::endl; // 5*

Здесь значение объекта в динамической памяти равно 5.

Далее, используя указатель, можно присвоить динамическому объекту другое значение:

*int \*ptr{new int{5}};*

*std::cout << "\*ptr = " << \*ptr << std::endl; // \*ptr = 5*

*\*ptr = 22;*

*std::cout << "\*ptr = " << \*ptr << std::endl; // \*ptr = 22*

**3 Освобождение памяти**

Динамические объекты будут существовать, пока не будут явным образом удалены. И после завершения использования динамических объектов следует освободить их память с помощью оператора delete:

*#include <iostream>*

*int main()*

*{*

*int \*ptr{new int{5}}; // выделяем память*

*std::cout << "\*ptr = " << \*ptr << std::endl; // \*ptr = 5*

*delete ptr; // освобождаем память*

*}*

Особенно это надо учитывать, если динамический объект создается в одной части кода, а используется в другой. Например:

*#include <iostream>*

*int\* createPtr(int value)*

*{*

*int \*ptr {new int{value}};*

*return ptr;*

*}*

*void usePtr()*

*{*

*int \*obj = createPtr(10);*

*std::cout << \*obj << std::endl; // 10*

*delete obj; // объект надо освободить*

*}*

*int main()*

*{*

*usePtr();*

*}*

В функции usePtr() получаем из функции createPtr() указатель на динамический объект. Однако после выполнения функции usePtr() этот объект автоматически не удаляется из памяти (как это происходит в случае с локальными автоматическими объектами). Поэтому его надо явным образом удалить, использовав оператор delete.

Если явным образом не вызвать оператор delete, то выделенная динамическая память будет освобождена после завершения программы.

Даже после освобождения памяти указатель по-прежнему содержит старый адрес, хотя память по нему условно освобождена и готова к использованию для будущих динамических объектов. Такой указатель называется "болтающимся указателем" (dangling pointer). Можно попробовать обратиться по этому указателю. Но использование объекта по указателю после его удаления или повторное применение оператора delete к указателю могут привести к непредсказуемым результатам:

*int \*ptr {new int{12}};*

*std::cout << \*ptr << std::endl; // 12*

*delete ptr;*

*// ошибочные сценарии*

*std::cout << \*ptr << std::endl; // объект по указателю ptr уже удален!*

*delete ptr; // объект по указателю ptr уже удален!*

Чтобы избежать подобных болтающихся указателей, рекомендуется после освобождения памяти обнулять указатель:

*int \*ptr {new int{12}};*

*std::cout << \*ptr << std::endl; // 12*

*delete ptr;*

*ptr = nullptr; // обнуляем указатель*

При попытке обращения к объекту через нулевой указатель программа просто завершит выполнение. А применение оператора delete к нулевому указателю не имеет никакого эффекта.

Также нередко имеет место ситуация, когда на один и тот же динамический объект указывают сразу несколько указателей. Если оператор delete применен к одному из указателей, то память объекта освобождается, и по второму указателю этот объект мы использовать уже не сможем. Если же после этого ко второму указателю применить оператор delete, то динамическая память может быть нарушена.

В то же время недопустимость указателей после применения к ним оператора delete не означает, что эти указатели мы в принципе не сможем использовать. Мы сможем их использовать, если присвоим им адрес другого объекта:

*#include <iostream>*

*int main()*

*{*

*int \*p1 {new int{12}};*

*int \*p2 {p1}; // p1 и p2 указывают на один и тот же объект*

*std::cout << \*p1 << std::endl; // 12*

*std::cout << \*p2 << std::endl; // 12*

*delete p1; // адреса в p1 и p2 недопустимы*

*p1 = nullptr;*

*p2 = nullptr;*

*p1 = new int{11}; // p1 указывает на новый объект*

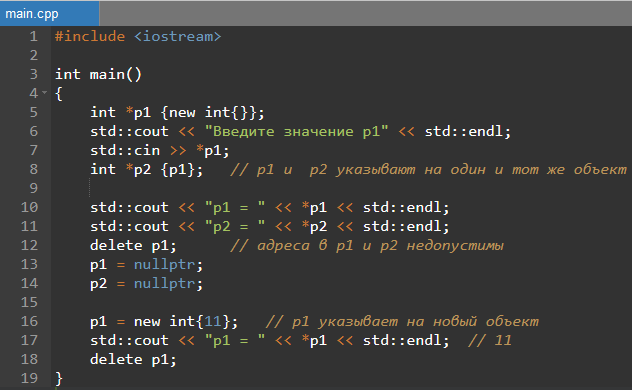
*std::cout << \*p1 << std::endl; // 11*

*delete p1;*

*}*

Здесь после удаления объекта, на который указывает p1, этому указателю передается адрес другого объекта в динамической памяти. Соответственно мы также можем использовать указатель p1. В то же время адрес в указателе p2 по прежнему будет недействительным.

**Задача 8**. Объявить указатель на переменную типа *int*. Ввести ее значение с клавиатуры. Объявить второй указатель на переменную того же типа. Передать второму указателю адрес 1-го объекта. Вывести оба значения типа *int*. Удалить адрес 1-го объекта. Присвоить 1-му объекту новое значение.



Консоль:

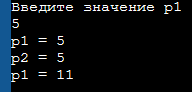


Таблица 8 – Варианты заданий для задачи № 8

|  |  |
| --- | --- |
| № варианта | Типы переменных |
| 1 | float |
| 2 | long double |
| 3 | double |
| 4 | long |
| 5 | long long |
| 6 | unsigned long |
| 7 | unsigned long long |
| 8 | unsigned int |
| 9 | float |
| 10 | long double |
| 11 | double |
| 12 | long |
| 13 | long long |
| 14 | unsigned long |
| 15 | unsigned long long |
| 16 | unsigned int |
| 17 | float |
| 18 | long double |
| 19 | double |
| 20 | long |
| 21 | long long |
| 22 | unsigned long |
| 23 | unsigned long long |
| 24 | unsigned int |
| 25 | float |
| 26 | long double |
| 27 | double |
| 28 | long |
| 29 | long long |
| 30 | unsigned long |
| 31 | unsigned long long |
| 32 | unsigned int |

**4 Динамические массивы**

Кроме отдельных динамических объектов в языке C++ можно использовать динамические массивы. Для выделения памяти под динамический массив также используется оператор **new**, после которого в квадратных скобках указывается, сколько массив будет содержать объектов:

*int \*numbers {new int[4]}; // динамический массив из 4 чисел*

*// или так*

*// int \*numbers = new int[4];*

Причем в этом случае оператор new также возвращает указатель на объект типа int - первый элемент в созданном массиве.

В данном случае определяется массив из четырех элементов типа int, но каждый из них имеет неопределенное значение. Однако можно инициализировать массив значениями:

*int \*numbers1 {new int[4]{}}; // массив состоит из чисел 0, 0, 0, 0*

*int \*numbers2 {new int[4]{ 1, 2, 3, 4 }}; // массив состоит из чисел 1, 2, 3, 4*

*int \*numbers3 {new int[4]{ 1, 2 }}; // массив состоит из чисел 1, 2, 0, 0*

*// аналогичные определения массивов*

*// int \*numbers1 = new int[4]{}; // массив состоит из чисел 0, 0, 0, 0*

*// int \*numbers1 = new int[4](); // массив состоит из чисел 0, 0, 0, 0*

*// int \*numbers2 = new int[4]{ 1, 2, 3, 4 }; // массив состоит из чисел 1, 2, 3, 4*

*// int \*numbers3 = new int[4]{ 1, 2 }; // массив состоит из чисел 1, 2, 0, 0*

При инициализации массива конкретными значениями следует учитывать, что если значений в фигурных скобках больше чем длина массива, то оператор new потерпит неудачу и не сможет создать массив. Если переданных значений, наоборот, меньше, то элементы, для которых не предоставлены значения, инициализируются значением по умолчанию.

В стандарт С++20 добавлена возможность выведения размера массива, поэтому, если применяется стандарт С++20, то можно не указывать длину массива:

*int \*numbers {new int[]{ 1, 2, 3, 4 }}; // массив состоит из чисел 1, 2, 3, 4*

После создания динамического массива мы сможем с ним работать по полученному указателю, получать и изменять его элементы:

*int \*numbers {new int[4]{ 1, 2, 3, 4 }};*

*// получение элементов через синтаксис массивов*

*std::cout << numbers[0] << std::endl; // 1*

*std::cout << numbers[1] << std::endl; // 2*

*// получение элементов через операцию разыменования*

*std::cout << \*numbers << std::endl; // 1*

*std::cout << \*(numbers+1) << std::endl; // 2*

Причем для доступа к элементам динамического массива можно использовать как синтаксис массивов (numbers[0]), так и операцию разыменования (\*numbers).

Соответственно для перебора такого массива можно использовать различные способы:

*unsigned n{ 5 }; // размер массива*

*int\* p{ new int[n] { 1, 2, 3, 4, 5 } };*

*// используем индексы*

*for (unsigned i{}; i < n; i++)*

*{*

*std::cout << p[i] << "\t";*

*}*

*std::cout << std::endl;*

*// добавляем к адресу в указателе смещение*

*for (unsigned i{}; i < n; i++)*

*{*

*std::cout << \*(p+i)<< "\t";*

*}*

*std::cout << std::endl;*

*// проходим по массиву с помощью вспомогательного указателя*

*for (int\* q{ p }; q != p + n; q++)*

*{*

*std::cout << \*q << "\t";*

*}*

*std::cout << std::endl;*

Для задания размера динамического массива мы можем применять обычную переменную, а не константу, как в случае со стандартными массивами.

Для удаления динамического массива и освобождения его памяти применяется специальная форма оператора delete:

*delete [] указатель\_на\_динамический\_массив;*

Например:

*#include <iostream>*

*int main()*

*{*

*unsigned n{ 5 }; // размер массива*

*int\* p{ new int[n] { 1, 2, 3, 4, 5 } };*

*// используем индексы*

*for (unsigned i{}; i < n; i++)*

*{*

*std::cout << p[i] << "\t";*

*}*

*std::cout << std::endl;*

*delete [] p;*

*}*

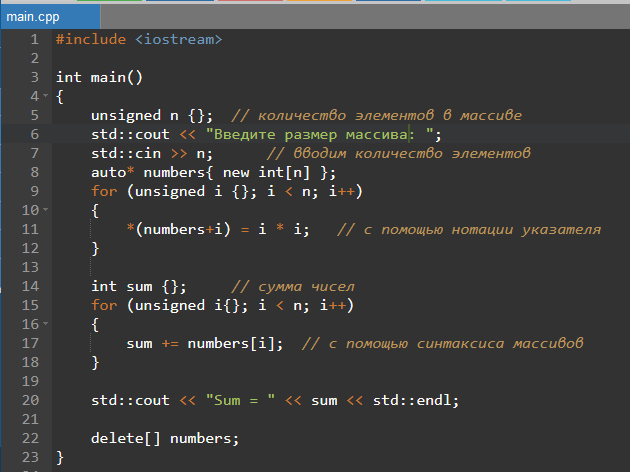
Чтобы после освобождения памяти указатель не хранил старый адрес, также рекомендуется обнулить его:

*delete [] p;*

*p = nullptr; // обнуляем указатель*

**Задача 9**. Напишите программу, в которой пользователь вводит размер массива, и программа динамически выделяет массив такого размера для хранения значений типа *int*. Используя указатель, инициализируйте все элементы массива так, чтобы значение элемента по индексу *i* было равно *i* \* *i* (то есть квадрату числа *i*). Вычислите сумму элементов, используя синтаксис массивов (обращение к элементам по индексу в квадратных скобках), и выведите результат на консоль.

Решение задачи 9.



Консоль:



Ключевое **auto** слово указывает компилятору использовать выражение инициализации объявленной переменной или параметра лямбда-выражения для определения ее типа. Синтаксис:

*auto ИнициализаторДекларатора;*

*[] (auto param1, auto param2) {};*

Таблица 9 – Варианты заданий для задачи № 9

|  |  |
| --- | --- |
| № варианта | Задание |
| 1 | Подсчитать количество отрицательных элементов массива |
| 2 | Подсчитать количество положительных элементов массива |
| 3 | Подсчитать количество элементов массива, кратных 3 |
| 4 | Подсчитать количество элементов массива, кратных 4 |
| 5 | Подсчитать количество элементов массива, принадлежащих диапазону значений a < mas[i] < b |
| 6 | Подсчитать количество неотрицательных элементов массива |
| 7 | Подсчитать количество элементов массива, кратных 5 |
| 8 | Подсчитать количество элементов массива, принадлежащих диапазону значений (a < mas[i] < b) || (c < mas[i] < d) |
| 9 | Подсчитать количество отрицательных элементов массива |
| 10 | Подсчитать количество положительных элементов массива |
| 11 | Подсчитать количество элементов массива, кратных 3 |
| 12 | Подсчитать количество элементов массива, кратных 4 |
| 13 | Подсчитать количество элементов массива, принадлежащих диапазону значений a < mas[i] < b |
| 14 | Подсчитать количество неотрицательных элементов массива |
| 15 | Подсчитать количество элементов массива, кратных 5 |
| 16 | Подсчитать количество элементов массива, принадлежащих диапазону значений (a < mas[i] < b) || (c < mas[i] < d) |
| 17 | Подсчитать количество отрицательных элементов массива |
| 18 | Подсчитать количество положительных элементов массива |
| 19 | Подсчитать количество элементов массива, кратных 3 |
| 20 | Подсчитать количество элементов массива, кратных 4 |
| 21 | Подсчитать количество элементов массива, принадлежащих диапазону значений a < mas[i] < b |
| 22 | Подсчитать количество неотрицательных элементов массива |
| 23 | Подсчитать количество элементов массива, кратных 5 |
| 24 | Подсчитать количество элементов массива, принадлежащих диапазону значений (a < mas[i] < b) || (c < mas[i] < d) |
| 25 | Подсчитать количество отрицательных элементов массива |
| 26 | Подсчитать количество положительных элементов массива |
| 27 | Подсчитать количество элементов массива, кратных 3 |
| 28 | Подсчитать количество элементов массива, кратных 4 |
| 29 | Подсчитать количество элементов массива, принадлежащих диапазону значений a < mas[i] < b |
| 30 | Подсчитать количество неотрицательных элементов массива |
| 31 | Подсчитать количество элементов массива, кратных 5 |
| 32 | Подсчитать количество элементов массива, принадлежащих диапазону значений (a < mas[i] < b) || (c < mas[i] < d) |

**5 Многомерные массивы**

Можно создавать многомерные динамические массивы. Рассмотрим на примере двухмерных массивов. Двухмерный массив – это набор массив массивов. Соответственно, чтобы создать динамический двухмерный массив, нужно создать общий динамический массив указателей, а затем его элементы - вложенные динамические массивы. В общем случае это выглядит так:

*#include <iostream>*

*int main()*

*{*

*unsigned rows = 3; // количество строк*

*unsigned columns = 2; // количество столбцов*

*int\*\* numbers{new int\*[rows]{}}; // выделяем память под двухмерный массив*

*// выделяем память для вложенных массивов*

*for (unsigned i{}; i < rows; i++)*

*{*

*numbers[i] = new int[columns]{};*

*}*

*// удаление массивов*

*for (unsigned i{}; i < rows; i++)*

*{*

*delete[] numbers[i];*

*}*

*delete[] numbers;*

*}*

Вначале выделяем память для массива указателей (условно таблицы):

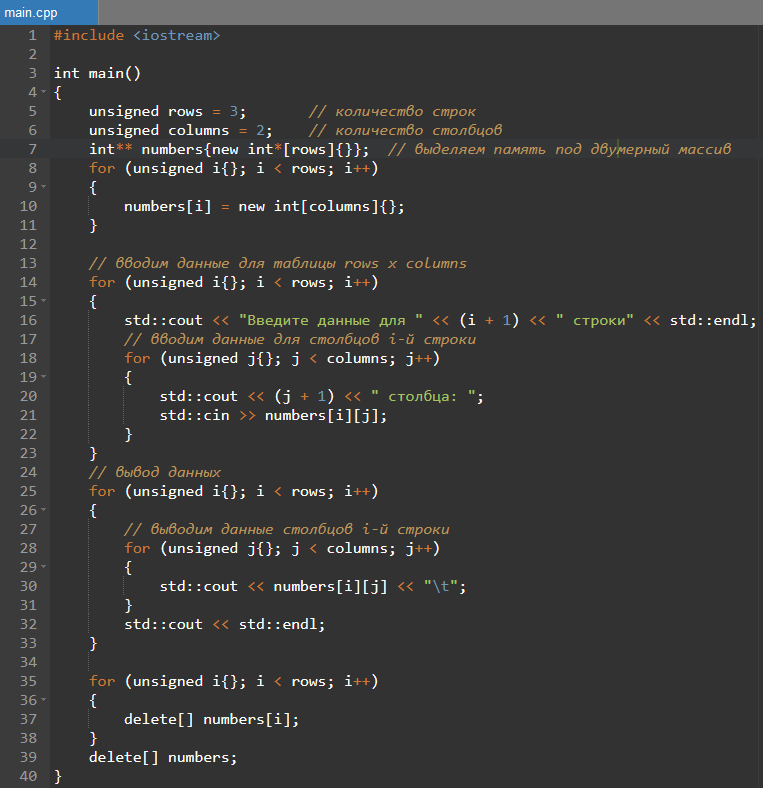
*int\*\* numbers{new int\*[rows]{}};*

Затем в цикле выделяем память для каждого отдельного массива (условно строки таблицы):

*numbers[i] = new int[columns]{};*

Освобождение памяти идет в обратном порядке - сначала освобождаем память для каждого отдельного вложенного массива, а затем для всего массива указателей.

**Задача 10.** Выделить память под двумерный массив. Ввести значения с клавиатуры. Вывести значения на экран.



Консоль:

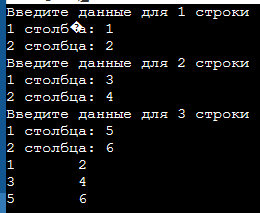


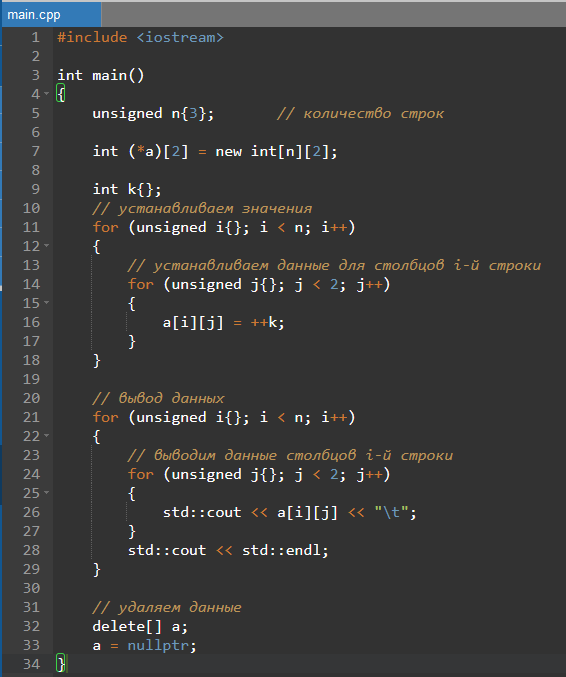
Таблица 10 – Варианты заданий для задачи № 10

|  |  |
| --- | --- |
| № варианта | Доп. задание |
| 1 | Найти максимальный диагональный элемент и указать его координаты в главной диагонали матрицы |
| 2 | Подсчитать количество нечетных элементов в каждой строке двумерного массива |
| 3 | Во всех строках двумерного массива заменить максимальные элементы нулем |
| 4 | Вычислить произведение элементов двумерного массива |
| 5 | Определить координаты максимального по величине элемента главной диагонали двумерного массива |
| 6 | В двумерном массиве определить количество положительных и отрицательных элементов |
| 7 | Определить число положительных элементов двумерного массива |
| 8 | Найти сумму элементов двумерного массива, значения которых положительны и четны |
| 9 | Найти максимальный диагональный элемент и указать его координаты в главной диагонали матрицы |
| 10 | Подсчитать количество нечетных элементов в каждой строке двумерного массива |
| 11 | Во всех строках двумерного массива заменить максимальные элементы нулем |
| 12 | Вычислить произведение элементов двумерного массива |
| 13 | Определить координаты максимального по величине элемента главной диагонали двумерного массива |
| 14 | В двумерном массиве определить количество положительных и отрицательных элементов |
| 15 | Определить число положительных элементов двумерного массива |
| 16 | Найти сумму элементов двумерного массива, значения которых положительны и четны |
| 17 | Найти максимальный диагональный элемент и указать его координаты в главной диагонали матрицы |
| 18 | Подсчитать количество нечетных элементов в каждой строке двумерного массива |
| 19 | Во всех строках двумерного массива заменить максимальные элементы нулем |
| 20 | Вычислить произведение элементов двумерного массива |
| 21 | Определить координаты максимального по величине элемента главной диагонали двумерного массива |
| 22 | В двумерном массиве определить количество положительных и отрицательных элементов |
| 23 | Определить число положительных элементов двумерного массива |
| 24 | Найти сумму элементов двумерного массива, значения которых положительны и четны |
| 25 | Найти максимальный диагональный элемент и указать его координаты в главной диагонали матрицы |
| 26 | Подсчитать количество нечетных элементов в каждой строке двумерного массива |
| 27 | Во всех строках двумерного массива заменить максимальные элементы нулем |
| 28 | Вычислить произведение элементов двумерного массива |
| 29 | Определить координаты максимального по величине элемента главной диагонали двумерного массива |
| 30 | В двумерном массиве определить количество положительных и отрицательных элементов |
| 31 | Определить число положительных элементов двумерного массива |
| 32 | Найти сумму элементов двумерного массива, значения которых положительны и четны |

**6 Указатель на массив**

От типа **int\*\***, который представляет указатель на указатель (pointer-to-pointer) следует отличать ситуацию "указатель на массив" (pointer to array).

**Задача 11.** Осуществить инициализации данными элементов двумерного массива. Вывести значения элементов.



Консоль:



Здесь запись **int (\*a)[2]** представляет указатель на массив из двух элементов типа int. Фактически можно работать с этим объектом как с двухмерным массивом (таблицей), только количество столбцов в данном случае фиксировано - 2. И память для такого массива выделяется один раз:

*int (\*a)[2] = new int[n][2];*

То есть в данном случае имеется таблица из n строк и 2 столбцов. Используя два индекса (для строки и столца), можно обращаться к определенному элементу, установить или получить его значение.

Таблица 11 – Варианты заданий для задачи № 11

|  |  |
| --- | --- |
| № варианта | Доп. задание |
| 1 | Дана целочисленная квадратная матрица *A(N,N)*.Найдите:  а) значение наименьшего из элементов главной диагонали (минимум); б) количество элементов матрицы, меньших чем значение минимума. |
| 2 | Дана целочисленная квадратная матрица A(N,N). Найдите:  а) значение наибольшего из элементов главной диагонали (максимум);  б) количество элементов матрицы, превышающих значение максимума. |
| 3 | Дана целочисленная квадратная матрица A(N,N). Найдите сумму элементов матрицы, расположенных в строках с отрицательным элементом на главной диагонали. |
| 4 | Дана целочисленная квадратная матрица A(N,N). Просматривая её элементы в заданном порядке, найдите первый чётный элемент и поменяйте его местами с диагональным элементом той строки, в которой он находится. Порядок просмотра: справа налево и снизу вверх. |
| 5 | Дана целочисленная квадратная матрица A(N,N). Просматривая её элементы в заданном порядке, найдите первый нечётный элемент и поменяйте его местами с диагональным элементом той строки, в которой он находится. Порядок просмотра: снизу вверх и слева направо. |
| 6 | Дана целочисленная квадратная матрица A(N,N). Просматривая её элементы в заданном порядке, найдите первый чётный элемент и поменяйте его местами с диагональным элементом той строки, в которой он находится. Порядок просмотра: сверху вниз и справа налево. |
| 7 | Дана целочисленная квадратная матрица A(N,N).Найдите количество неотрицательных элементов матрицы, расположенных под главной диагональю. |
| 8 | Дана целочисленная квадратная матрица A(N,N).Найдите количество нечётных элементов матрицы, расположенных над главной диагональю. |
| 9 | Дана целочисленная квадратная матрица *A(N,N)*.Найдите:  а) значение наименьшего из элементов главной диагонали (минимум); б) количество элементов матрицы, меньших чем значение минимума. |
| 10 | Дана целочисленная квадратная матрица A(N,N). Найдите:  а) значение наибольшего из элементов главной диагонали (максимум);  б) количество элементов матрицы, превышающих значение максимума. |
| 11 | Дана целочисленная квадратная матрица A(N,N). Найдите сумму элементов матрицы, расположенных в строках с отрицательным элементом на главной диагонали. |
| 12 | Дана целочисленная квадратная матрица A(N,N). Просматривая её элементы в заданном порядке, найдите первый чётный элемент и поменяйте его местами с диагональным элементом той строки, в которой он находится. Порядок просмотра: справа налево и снизу вверх. |
| 13 | Дана целочисленная квадратная матрица A(N,N). Просматривая её элементы в заданном порядке, найдите первый нечётный элемент и поменяйте его местами с диагональным элементом той строки, в которой он находится. Порядок просмотра: снизу вверх и слева направо. |
| 14 | Дана целочисленная квадратная матрица A(N,N). Просматривая её элементы в заданном порядке, найдите первый чётный элемент и поменяйте его местами с диагональным элементом той строки, в которой он находится. Порядок просмотра: сверху вниз и справа налево. |
| 15 | Дана целочисленная квадратная матрица A(N,N).Найдите количество неотрицательных элементов матрицы, расположенных под главной диагональю. |
| 16 | Дана целочисленная квадратная матрица A(N,N).Найдите количество нечётных элементов матрицы, расположенных над главной диагональю. |
| 17 | Дана целочисленная квадратная матрица *A(N,N)*.Найдите:  а) значение наименьшего из элементов главной диагонали (минимум); б) количество элементов матрицы, меньших чем значение минимума. |
| 18 | Дана целочисленная квадратная матрица A(N,N). Найдите:  а) значение наибольшего из элементов главной диагонали (максимум);  б) количество элементов матрицы, превышающих значение максимума. |
| 19 | Дана целочисленная квадратная матрица A(N,N). Найдите сумму элементов матрицы, расположенных в строках с отрицательным элементом на главной диагонали. |
| 20 | Дана целочисленная квадратная матрица A(N,N). Просматривая её элементы в заданном порядке, найдите первый чётный элемент и поменяйте его местами с диагональным элементом той строки, в которой он находится. Порядок просмотра: справа налево и снизу вверх. |
| 21 | Дана целочисленная квадратная матрица A(N,N). Просматривая её элементы в заданном порядке, найдите первый нечётный элемент и поменяйте его местами с диагональным элементом той строки, в которой он находится. Порядок просмотра: снизу вверх и слева направо. |
| 22 | Дана целочисленная квадратная матрица A(N,N). Просматривая её элементы в заданном порядке, найдите первый чётный элемент и поменяйте его местами с диагональным элементом той строки, в которой он находится. Порядок просмотра: сверху вниз и справа налево. |
| 23 | Дана целочисленная квадратная матрица A(N,N).Найдите количество неотрицательных элементов матрицы, расположенных под главной диагональю. |
| 24 | Дана целочисленная квадратная матрица A(N,N).Найдите количество нечётных элементов матрицы, расположенных над главной диагональю. |
| 25 | Дана целочисленная квадратная матрица *A(N,N)*.Найдите:  а) значение наименьшего из элементов главной диагонали (минимум); б) количество элементов матрицы, меньших чем значение минимума. |
| 26 | Дана целочисленная квадратная матрица A(N,N). Найдите:  а) значение наибольшего из элементов главной диагонали (максимум);  б) количество элементов матрицы, превышающих значение максимума. |
| 27 | Дана целочисленная квадратная матрица A(N,N). Найдите сумму элементов матрицы, расположенных в строках с отрицательным элементом на главной диагонали. |
| 28 | Дана целочисленная квадратная матрица A(N,N). Просматривая её элементы в заданном порядке, найдите первый чётный элемент и поменяйте его местами с диагональным элементом той строки, в которой он находится. Порядок просмотра: справа налево и снизу вверх. |
| 29 | Дана целочисленная квадратная матрица A(N,N). Просматривая её элементы в заданном порядке, найдите первый нечётный элемент и поменяйте его местами с диагональным элементом той строки, в которой он находится. Порядок просмотра: снизу вверх и слева направо. |
| 30 | Дана целочисленная квадратная матрица A(N,N). Просматривая её элементы в заданном порядке, найдите первый чётный элемент и поменяйте его местами с диагональным элементом той строки, в которой он находится. Порядок просмотра: сверху вниз и справа налево. |
| 31 | Дана целочисленная квадратная матрица A(N,N).Найдите количество неотрицательных элементов матрицы, расположенных под главной диагональю. |
| 32 | Дана целочисленная квадратная матрица A(N,N).Найдите количество нечётных элементов матрицы, расположенных над главной диагональю. |