**Программирование на С++**

**9-11 классы**

**Занятие 6**

**Тема: Статическая и динамическая память**

**Цели и задачи:**

* Научиться работать с динамической памятью
* Научиться работать с двойным указателем
* Научиться грамотному распределению ресурсов памяти

**По результатам занятия слушатель будет знать:**

* В чём разница использования статической и динамической памяти
* Что такое указатель на указатель.
* Что такое двумерный массив
* Что такое матрица

**По результатам занятия слушатель будет уметь:**

* Создавать матрицы
* Создавать поля и методы класса

**Теоретическая часть**

**Виды памяти «внутри» компьютера**

В программах на языке С++, память разделена на 2 части:

**Стек (stack):** все локальные переменные размещаются в памяти стека.

**Куча (heap):** Неиспользованная программой память, которая может быть использована, когда программа **динамически** выделяет память.

Часто, вы не будете знать сколько памяти вам понадобится для хранения информации в определённых переменных и объём необходимой памяти будет определён при запуске программы.

В таком случае вы можете выделить память во время выполнения программы с помощью кучи для переменных заданного типа используя оператор **new**, который возвращает адрес выделенной памяти.

**Работа с адресами и указателями**

Выделенный адрес может быть сохранён в указателе, который в последствии может быть разадресован (разименован) для доступа к переменной (её значению)

Имеем динамически выделенную память для целых чисел, а затем ей присвоено значение 5.

Указатель p хранится в стеке как локальная переменная и хранит выделенный в куче адрес. Значение 5 хранится по этому адресу в куче

**Стек**

Для локальных переменных в стеке, управление памятью осуществляется автоматически. В куче, необходимо вручную управлять динамически выделенной памятью и использовать оператор delete для освобождения памяти, когда в ней больше нет необходимости.

Если забывать освобождать выделенную (динамически) память, то образуются утечки памяти, потому что эта память остаётся выделенной, пока программа не будет закрыта.

Оператор **delete** освобождает память, выделенную для переменной, но не удаляет сам указатель (который хранит только адрес объекта), т к сам указатель хранится в стеке.

Указатель NULL - это константа со значением равным нулю, которая определена в нескольких стандартных библиотеках.

Присваивание NULL указателю при его объявлении является хорошей практикой, в случае если у вас нет точного адреса для присваивания.

**Динамическая память**

Динамическая памятьтакже может быть выделена для массива

Динамическое выделение памяти полезно во многих случаях, к примеру, когда вашей программе необходимо получить изображение, но она не знает его возможный размер и количество памяти, необходимое для его хранения

**Многомерные массивы**

Многомерный массив хранит один и более массивов

Например, объявление двумерного массива целых чисел выглядит следующим образом.

**int x[3][4];**

Такую конструкцию удобно представлять себе в виде таблицы, состоящей из 3 строк и 4 столбцов.

(Помните, что индексирование массивов начинается с 0)

Многомерные массивы могут быть инициализированы с использованием заключения внутри специальных скобок для каждой строки

**Случайные числа (Псевдослучайные)**

Возможность генерировать **случайные** числа очень полезна во многих ситуациях, включая создания игр, программ статического моделирования и подобных продуктов.

В стандартной библиотеке С++ определена функция **rand()**, которая возвращает псевдослучайное число. Для её использования необходимо с помощью директивы (инструкции) препроцессора подключить заголовочный файл **<cstdlib>**

**Практическая часть**

**Задание 1**

Создайте матрицу (двумерный массив) 5 на 5, выделив для него память динамически. Заполните все элементы нулями и выведите на экран

\*

Вынесете в отдельную функцию вывод матрицы (двумерного массива) на экран (стандартный вывод), выделение памяти для матрицы и присваивание случайных значений элементам матрицы

\*\*  
Напишите функцию сложения двух матриц (двумерных массивов)

Каждый элемент первого массива складывается с соответствующим элементом (имеющим такие же индексы) второго массива.

\*\*\*

Написать функцию, контролирующую утечки памяти (удаляющую динамически выделенную память)

(Файл с кодом лежит в папке code/)