Referencias	Aprendizajes	Preguntas/comentarios
https://www.geeksforgeeks.org/how-to-declare-a-2d-array-dynamically-in-c-using-new-operator/?ref=gcse	Hay 2 formas de declarar arreglos de 2d usando memoria dinamica: 1. Usando un puntero simple: // Dimensiones int m = 3, n = 4, c = 0; // Separando un espacio de memoria en el heap para las filas int* arr = new int[m * n]; // Separando espacio de memoria en el heap para las columnas for (int i = 0; i < m; i++) { for (int j = 0; j < n; j++) // Asignando valores a los bloques de memoria *(arr + i * n + j) = ++c; } 2. Usando un arreglo de punteros: // Dimensiones int m = 3, n = 4, c = 0; // Separando un espacio de memoria en el heap para las filas int** a = new int*[m]; for (int i = 0; i < m; i++) { //Separando espacio de memoria en el heap para las columnas a[i] = new int[n]; } //Asignando valores a los bloques de memoria for (int i = 0; i < m; i++) { for (int j = 0; j < n; j++) a[i][j] = ++c; }	En lo personal me gusto más el arreglo de punteros

https://www.geeksforgeeks.org/how-to-dynamically-allocate-a-3d-array-in-c/?ref=gcse	Igual que con 2d hay dos metodos, pero en este caso se tiene en cuenta cada dimesión de esta manera: X = # de matrices 2d Y =# de filas de cada matriz 2d Z = # de columnas de cada matriz 2d 1. ``` int x = 2, y = 3, z = 4; int count = 0; int* a = new int[x * y * z]; for (int i = 0; i < x; i++) { for (int j = 0; j < y; j++) { for (int k = 0; k < z; k++) { *(a + i * y * z + j * z + k) = ++count; } } } ``` 2. ``` int x = 2, y = 3, z = 4; int count = 0; int*** a = new int**[x]; for (int i = 0; i < x; i++) { //Separando memoria en el heap para las filas a[i] = new int*[y]; for (int j = 0; j < y; j++) //Separando memoria en el heap para las columnas a[i][j] = new int[z]; } for (int i = 0; i < x; i++) { for (int j = 0; j < y; j++) { for (int k = 0; k < z; k++) a[i][j][k] = ++count; } } ```	En lo personal me gusto más el arreglo de punteros

https://www.geeksforgeeks.org/new-and-delete-operators-in-cpp-for-dynamic-memory/?ref=gcse		
https://www.geeksforgeeks.org/multidimensional-pointer-arithmetic-in-cc/?ref=gcse		
https://www.geeksforgeeks.org/array-of-structures-vs-array-within-a-structure-in-c-and-cpp/?ref=gcse	Aprendimos que es una estructura en c++, es un tipo de dato que permite tratar un grupo de variables relacionadas como una sola unidad, en lugar de entidades separadas. Una estructura puede contener elementos de diferentes tipos de datos, incluso una matriz. // Declarando una estructura llamada candidate **struct** candidate { int roll_no; char grade; // Matriz dentro de una estructura float marks[4]; }; // Función para mostrar/imprimir el contenido de la estructura void display(struct candidate a1) { cout << "Roll number : " << a1.roll_no << endl; cout <<"Grade : " << a1.grade << endl; cout << "Marks secured:\n"; int i; int len = sizeof(a1.marks) / sizeof(float); // Accediento al contenido de la matriz for (i = 0; i < len; i++) cout << "Subject " << i + 1 << " : " << a1.marks[i] << endl; } int main() { // Inicializando la estructura **struct** candidate A= { 1, 'A', { 98.5, 77, 89, 78.5 } }; display(A); // imprimiendo la variable A tipo struct candidate return 0; }	¿El orden en que se inicializa la instancia de la estructura depende del orden en que hallamos creado las variables dentro de la estructura?

https://stackoverflow.com/questions/18469859/c-multidimensional-array-multiple-data-types		¿cuando se usa structure pair, pair es una palabra reservada?
https://www.geeksforgeeks.org/pair-in-cpp-stl/?ref=gcse		Es una buena herramienta, no es dificl de usar o entender, hace parte de las STL. Pero... ¿Sólo sirve para almacenar 2 datos? Por otro lado tiene ya funciones incluidas que facilitan varias cosas como, swap que es trucar los valores de pair1 con pair2.