Task 0 part b 256 is 0x100h

Addresses in decimal

The top Layer shows the 3d array base address while the inner layers inside the top array shows that there are 3 2d array with 4 element each array has.

We are Looking for the Array[2][1][3]

256	**260**	**264**	**268**	**272**	**276**	**280**	**284**	**288**	**292**	**296**	**300**
304	308	312	316	320	324	328	332	336	340	344	348
352	356	360	364	368	372	376	380	384	388	392	396
400	404	408	412	416	420	424	428	432	436	440	444
448	452	456	460	464	468	472	476	480	484	488	492
496	500	504	508	512	516	520	524	528	532	536	540

Which means array at index 2 inside it's first 2d array and it's 3rd element which is 380 in decimal. So it's at 380 which is 17C in the hex decimal

Array[2][1][3]

Calculations

We have Student[6][3][4]

And we have to find address of [2][1][3]

Then we can calculate it using row wise or column wise but am gonna do it in row wise.

Remember, am using decimal & will convert answer to the hexa.

Address = B.A + sizeof(type)*(total_row*total_col (desired_array_index) + total_col(desired_row_index) + desired_col_index)

Address = 256 + sizeof(int)*(((3*4)(2) + 4(1) + 3)

Suppose size of int is 4

Address = 256 + 4*(12(2)+4+3)

Address= 256 + 4(31)

Address = 380

In Hexadecimal 380 is 0x17C

Task 0 Part A

const int City = 6, School = 3, Class = 4;

int enrolled_Students [City][School][Class];

enrolled_Students [2][1][3] represents that there are two city which have 1 school (each city) and each school has 3 Classes in it.

In technical terms it means there are 2 2D arrays in such a way that each 2d array has 3 elements. I have explained the diagram above in task b to have better understanding.

Task 01

In this task I made sure there is input validation involved when taking input from the user. There are two functions in the program One of them is for creating the array while the other one is for the calculation of the average of the elements of the array entered by the user.

Task 02

In this task I made 3 pointers to take input from the user. First matrix entered by the user will be displayed row wise and column wise to the user. There are different functions for creating dynamic array, row wise display and col wise display. After that I ask the user for the entrance of the 2^{nd} & 3^{rd} matrix to multiply. I made sure condition of the multiplied is satisfied. In the end, multiplied matrix will be shown to the user by using mulMat function.

Task 03

First of all, I took the number of array's user wants to sort. Then that input (first array) is stored in a dynamic array by using a function. Then Another function is called from the main. IF user has just entered one array, then we call that function & take input from the user for the second array. After that using a trick, I sort the two arrays. like if element of first array is compared to the element of the second array as both arrays are in ascending so I took its advantage. First array element (which is bigger than being looped element of second array) then their position is replaced & second array is sorted. Then the size of the final array is calculated and the same function is called again to compare the next two array. In simple words, two arrays are sorted then they are appended to a new array & then I remove duplicates & made them descending from ascending order.

Task 04

In this task I made six arrays' according to question & filled them with data from the user in such a way that each index corresponds to the particular person data. (I mean index 0 of all arrays contain data about one person). Then I calculated the payrate & wages according to formula and filled them too in their correct position. For showing the top client by filtering I used a template function that takes the array and return the maximum index of particular person then that filtered array (person data) is shown to the User.

Task 05

For the game of life, first I made random function & filled the 2d array. I made the copy of this array too. Then I added the infinite loop that Loop for each cell & scan the copied array for the next upcoming array (next generation). There is a function in the loop that returns the number of alive cells around each cell of the 2d array then those number of alive cells are checked for weather the next cells should be created as alive or dead.