

How to declare a pointer to a 2d float matrix?

Asked 8 years, 2 months ago Active 1 year, 5 months ago Viewed 5k times

Im trying to declare a pointer to a 2d float matrix in order to have a dynamical behaviour of my image data but Im having a compilation error C2057: expected constant expression. I thought a pointer had to be casted in that way but apparently not.. Please anyone could help me ? Thanks!!

```
//Image size input
int imheight;
int imwidth;

cout << "Please, enter image height: \n>";
scanf ("%d",&imheight);
cout << "Please, enter image width: \n>";
scanf ("%d",&imwidth);

const int imheight2 = imheight;
const int imwidth2 = imwidth;

float *ZArray[imheight2][imwidth2];
```

Here is one of my other functions where I'm trying to hace access to ZArray. Im not getting the data properly read:

```
void LoadRIS( char* inputFileName , float** ZArray, int imageHeight , int
imageWidth){
    // Load input RIS file
    FILE* lRis = fopen ( inputFileName, "rb" );

    // Jump to data position
    for (int i = 0; i < 88; i++){
        uchar a = getc (lRis);
    }

    // Read z array
    size_t counter = fread ( *ZArray , 1 , imageHeight * imageWidth *
sizeof(ZArray) , lRis );

    //Get max value of RIS
    float RISmax = ZArray [0][0];
    float RISmin = ZArray [0][0];
    for (int i=0; i<imageHeight; i++)
    {
        for (int j=0; j<imageWidth; j++)
        {
            if (ZArray[i][j] > RISmax)
                RISmax = ZArray [i][j];
            if (ZArray[i][j] < RISmin)
                RISmin = ZArray [i][j];
        }
    }

    std::cout<<"The max value of the RIS file is: "<<RISmax<<"\n";
    std::cout<<"The min value of the RIS file is: "<<RISmin<<"\n";
    Beep(0,5000);

    // Close input file
    fclose (lRis);
}
```

c++ pointers

Share Improve this question Follow

edited May 7 '13 at 11:07

asked May 6 '13 at 10:06

 Nicolai
333 ♦ 7 • 14

In addition to what the answers say, that type is a 2D array of pointers, not a pointer to a 2D array. You would want `float (*ZArray)[imheight2][imwidth2];`. – Joseph Mansfield May 6 '13 at 10:11

This question is tagged C and C++, but the answers are different. C has supported variable-length arrays for some time, so small variable-length arrays can and should be defined with `float foo[r][c];`. One of the tags should be deleted. – Eric Postpischil May 6 '13 at 11:43

7 Answers

Active Oldest Votes

Try this (dynamical allocation)

```
//Image size input
int imheight;
int imwidth;

cout << "Please, enter image height: \n>";
scanf ("%d",&imheight);
cout << "Please, enter image width: \n>";
scanf ("%d",&imwidth);

float** ZArray = new float*[imheight];
for(int i=0;i<imheight;i++){
    ZArray[i] = new float[imwidth];
}
```

```
for(int i=0;i<imheight;i++){
    delete[] zArray[i];
}
delete[] zArray;
```

Hope this helps :)

P.S. As @FrankH says, this calls too many `new`'s and `delete` s, wasting a lot of time. Better idea should be to alloc `imwidth*imheight` space together.

Share Improve this answer Follow

edited May 8 '13 at 1:46

answered May 6 '13 at 10:35



Working great, thanks! by the way, how can I access the values of the double pointer `float**` from another function? It's crashing – Nicolai May 7 '13 at 8:54

@Nicolai you mean access the values in the 2d array? Just using `zArray[i][j]` is ok. i would like to know about the whole code, could you edit the question and paste your 'another function' code? – hongtao May 7 '13 at 9:54

Hi hongtao, just pasted one function in the question. Im not getting a crash now, but I'm not able to read the file properly and load it into my `zArray` – Nicolai May 7 '13 at 11:08

ouch. `imheight+1` calls to `new`, for no better reason than to use a `array[x][y]` syntax. The error-prone `delete` loop (you're not storing the size of the '2D array' alongside it so need a global variable ...). Not everything that "technically works" is an answer. At the very very least, use only two `new / delete[]` calls, `float *tmp = new float[imwidth*imheight]; for(i=0;i<imheight;i++,tmp+=imwidth) zArray[i]=tmp; ... delete[] zArray[0]; delete[] zArray` in order to remove the need to "know" the dimensions on `delete`. I'll never understand why there are so many refs to this wart ... – FrankH. May 7 '13 at 23:20

```
const int imheight2 = imheight;
const int imwidth2 = imwidth;
```

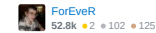
2

It doesn't make constant expressions. You cannot create array with such bounds. You should use `dynamic-allocation`, or `vector`.



Share Improve this answer Follow

answered May 6 '13 at 10:09



The problem is that you're declaring 2 `const int` variables but you're not assigning them `const` values. `imheight` and `imwidth` are not constant.

2

If you're fine with STL:

```
std::vector<std::valarray<float>> > floatMatrix;
```



edit: Just for your information, the space I placed between the `>` in the above line of code has nothing to do with my coding style. Your compiler might assume that `>>` is the right shift operator instead of 2 template argument list terminators. Angew's comment below sums it up.

Share Improve this answer Follow

edited May 6 '13 at 10:25

answered May 6 '13 at 10:09



1 The "old" interpretation of `>>` is not a bug, C++03 was simply defined that way (although many compilers supported treating `>>` as `> >` as an extension). First C++11 made this parse legal, so now it's mandatory for compilers to understand `>>` as template terminator when applicable. – Angew is no longer proud of SO May 6 '13 at 10:18

1 And seeing as it's supposed to be a matrix, I'd consider `std::valarray` instead of `std::vector`. – Angew is no longer proud of SO May 6 '13 at 10:19

I guess I can reword that last statement to be more general so I can avoiding misinforming anyone – user123 May 6 '13 at 10:20

@EricPostpischil: Indeed, but in C++ they can't (yet) be used to specify the size of an array, which is a problem. – Mike Seymour May 6 '13 at 11:47

@EricPostpischil: No it isn't. The problem is that the variables are `const` but their values are not constant expressions, hence can't be used to declare arrays. That's exactly what the first sentence says. – Mike Seymour May 6 '13 at 11:50

instead of `float *zArray[imheight2][imwidth2]`; it should be:

2

```
float **zArray = new float*[imheight2];
for(int i=0; i<imheight2; i++)
{
    zArray[i] = new float[imwidth2];
}
```



Share Improve this answer Follow

answered May 6 '13 at 10:33



If you have to do this, then code it at least as:

1

```
float **zArray = new float*[imheight];
float *tmp = new float[imheight*imwidth];
for(int i=0; i<imheight; i++, tmp += imwidth)
    zArray[i] = tmp;
```



This at least avoids doing more than two `new / delete[]` calls. And it preserves the functionality of your `fread(*zArray, ...)` which breaks if the memory isn't contiguous (and it won't generally be if you initialize this via many `new` calls).

A proper wrapper class would do just a single `new / malloc`, like:

```
template <class T> class Array2D {
private:
    size_t m_x;
    T* val;
public:
    Array2D(size_t x, size_t y) :
        m_x(x),
        val(new T[x*y]) {}
    ~Array2D() { delete[] val; }
    T* operator[](size_t y) { return val + y*m_x; }
}
```

You still cannot assign an instance of this to a `float**`. And it still allocates on the heap, where ordinary constant-dimension arrays can be on the stack. The only advantage of the additional allocation for the `float**` is that you're not bound to use a multiplication operation - but a separate memory access instead; that type of behaviour could be templated / traited into the wrapper class.

Generically, I'm more on the side of [multidimensional arrays are evil](#) (see also <https://stackoverflow.com/a/14276070/512360>, or [C++ FAQ.16.16](#)) but tastes vary ...

Share Improve this answer Follow

edited May 23 '17 at 11:50



Community ♦

1 • 1

answered May 7 '13 at 23:57



FrankH.

16.2k • 2 • 36 • 55

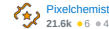
▲ You cannot use Arrays with dynamic sizes (your width and height variables are not compile time constant).

0 You can either use malloc() or new Operator to allocate Memory in a dynamic fashion.

▼ Share Improve this answer Follow

edited May 8 '13 at 9:27

answered May 6 '13 at 10:11



Pixelchemist

21.6k • 6 • 41 • 70

You're right but I think it is very likely that Nicolai is using C++, since variable-length arrays will not result in this compiler error C2057 in C. Therefore, this answer is correct with regard to the question. – [Pixelchemist](#) May 6 '13 at 12:17 ✓

That [plusplus.com](#) reference is no good ... even the code samples are buggy (the `FreeDynamicArray` leaks $(nRows - 1) * nCols$ units of `T`). It's just not right to use 'dynamic multidimensional arrays' in C/C++. Stick to `std::vector` and/or use/implement a proper `matrix` class - with `operator(int, int)` instead of `[] []`. – [FrankH.](#) May 8 '13 at 0:03

You are indeed right about that Website. I'll remove the link since this function may confuse People. But: Nobody should learn about how to use dynamic memory anymore, since we got the STL?! :X Everyone using C++ (especially when using C) SHOULD of course know about handling of one- or multi-dimensional arrays! Sooner or later you will be required to know such stuff even if you (try to) avoid using it in your own code. – [Pixelchemist](#) May 8 '13 at 9:26

You need to know about arrays, ack; and you need to understand that C/C++ "N-D arrays" are neither arrays nor matrices, in spite of the suggestively deceptive syntax. Also, in C/C++, "dynamic N-D arrays" and "statically-defined N-D arrays" can't be made into compatible types even if they've got the same dimensions. Corollary: if you need "N-D arrays", you really need a *matrix/tensor* class (else the multiple `new/delete` involved make dealing with errors very hard ... many `try {} catch {}` blocks). Speculating why the STL provides none may be a good topic for another question :) – [FrankH.](#) May 9 '13 at 10:06

- 1 I'm currently working on a project where we store N (where N changes during execution) vectors of size M (which does not change) to keep a system from returning to already visited states in an optimization process. This is in fact a NxM matrix of values where the number of rows constantly changes (vectors are added and removed during the execution). Contiguous storage would either require all values to be move/copied every time the size changes or a preallocation of very large memory chunks. We use vector-of-vectors but this is nothing but a wrapped "dynamic 2D array". – [Pixelchemist](#) May 12 '13 at 14:13

▲ `float *pMatrix = new float[imheight2*imwidth2];`

-1 then access elements like this

▼ `float f = pMatrix[x + imwidth2 * y];`

🕒 Share Improve this answer Follow

edited Feb 6 '20 at 6:56

answered May 6 '13 at 10:27



t.smith

89 • 1 • 1 • 7

- 1 I think you mean `y * width + x`. – [user123](#) May 6 '13 at 10:29