Filling a vector of pairs

Ask Question

I want to fill a vector with 8 pairs. Each pair represents the moves in x and y coordinates a knight in a game of chess can make. At the moment I'm doing it like this

```
vector<pair<int,int>> moves;

pair<int,int> aPair;
aPair.first = -2;
aPair.second = -1;
moves.push_back(aPair);
aPair.first = -2;
aPair.second = 1;
moves.push_back(aPair);
aPair.first = -1;
aPair.second = -2;
moves.push_back(aPair);
aPair.first = -1;
aPair.second = 2;
moves.push_back(aPair);
aPair.first = 1;
aPair.second = -2;
moves.push_back(aPair);
aPair.first = 1;
aPair.second = 2;
moves.push_back(aPair);
aPair.first = 2;
aPair.second = -1;
moves[6].push_back(aPair);
aPair.first = 2;
aPair.second = 1;
moves.push_back(aPair);
```

I'm doing this to learn about the Std library. This seems like a hopelessly inefficient way of solving this problem.

Anyone have a more elegant solution?

c++ stl

edited May 3 at 12:15

asked Nov 15 '12 at 21:46

Q-bertsuit
1,001 3 12 24

4 first observation: use `moves[0].push_back(std::make_pair(-2, -1));` second observation: You have 8 vectors not one ... andre Nov 15 '12 at

8 Answers

Loops to the rescue:

```
for(int k = 0; k < 2; k++)
    for(int i = -1; i < 2; i += 2)
        for(int j = -1; j < 2; j+=
            result.push_back(make_
```

Output: http://ideone.com/2B0F9b

answered Nov 15 '12 at 23:08

hate-engine
1,729 10 26

Wow that's a really nice solution. Thanks! – Q-bertsuit Nov 16 '12 at 19:22

If you have C++11 (otherwise you can't write `>>`), you can use the following:

```
vector<pair<int,int>> moves = {
  {-2, -1},
  {-2,  1},
  {-1, -2},
  {-1,  2},
  { 1, -2},
  { 1,  2},
  { 2, -1},
  { 2,  1}
};
```

answered Nov 15 '12 at 21:52

ipc
7,094 21 29

You need an additional pair of braces around each pair of numbers, the inner one performs aggregate initialization of the `std::pair` and the outer one is required for the `vector` constructor. – Praetorian Nov 15 '12 at 21:56

@Praetorian : `std::pair<>` is not an aggregate, that is a constructor call. – ildjarn Nov 15 '12 at 21:59

@ildjarn Hmm, always assumed it was. But gcc 4.7.0 is complaining

I don't have c++11, but thank you for posting! – Q-bertsuit Nov 16 '12 at 19:23

In C++98/03:

```
moves.push_back(std::make_pair(-2,
```

In C++11:

```
moves.emplace_back(-2, -1);
```

Alternatively in C++11:

```
std::vector<std::pair<int, int>> m
```

answered Nov 15 '12 at 21:58

Kerrek SB
347k 58 649 886

If you don't have C++11 you can utilize make_pair, pre-allocate the space for the vector without initializing the elements using reserve, and then utilize push_back without new allocations being done.

For example:

```
vector<pair<int,int> > moves;
moves.reserve(8);
moves.push_back(make_pair(-2, -1))
    // and so on
```

Even if you have C++11 this technique is useful if you need to compute the elements on the fly rather than hard code them.

edited Nov 15 '12 at 21:59

answered Nov 15 '12 at 21:54

Josh Heitzman
1,696 1 10 26

Thank you for this. I already marked an answer, but I think it should have gone her. – Q-bertsuit Nov 16 '12 at 19:24

You're welcome! – Josh Heitzman Nov 16 '12 at 20:01

Initializer list together with Uniform Initialization gives a lot of power in C++11.

answered Nov 15 '12 at 21:54

Mateusz Pusz
1,015 6 15

Here's another method of doing the same thing.

```
template <class VectorClass>
class CreateVector
{
public:
    typedef typename VectorClass::v
    CreateVector(const value_type&
    {
        mVector.push_back(value);
    }

    CreateVector& operator()(const
    {
        mVector.push_back(value);
        return *this;
    }

    inline operator VectorClass() c
    {
        return mVector;
    }
private:
    VectorClass mVector;
};
```

Usage:

```
vector<pair<int,int>> moves = Creat
(make_pair(1,2))
(make_pair(2,3))
(make_pair(3,4))
(make_pair(4,5));
```

EDIT: Provided you're not using C++11, this would be one way. Otherwise, I would suggest to go the way @ipc suggested.

answered Nov 15 '12 at 22:01

Vite Falcon
5,566 1 21 43

If you're using C++11, you might want to consider std::array instead of std::vector. Like a normal array, the std array has a fixed number of elements and makes more conceptual sense if you know in advance how much data you use.

answered Nov 15 '12 at 23:50

Alexander Duchene

with loops:

```
vector<pair<int, int>> result;
for(int moveX=1; moveX<=2; moveX++
{
    for(int signX=-1; signX<=1; si
    {
        for(int signY=-1; signY<=1
        {
            result.push_back(make_
        }
    }
}
```

Full program produces the following vector:

```
{-1, -2},
{-1, 2},
{1, -2},
{1, 2},
{-2, -1},
{-2, 1},
{2, -1},
{2, 1},
```

answered Jul 31 '15 at 8:26

Serge Rogatch
5,412 2 29 57