```
for (int i = 0; i < m_numShapeVertices; i++)
    {
        if (i >= 0 && i < slices) //Orders top faces
        {
            if (i == slices - 1)
            {
                vIndex[(currentFace * 3) + 0] = 0;
                vIndex[(currentFace * 3) + 1] = i;
                vIndex[(currentFace * 3) + 2] = m_numShapeVertices - 2;

                cout << "Face " << currentFace << " <" << vIndex[(currentFace * 3) + 0]
<< "," << vIndex[(currentFace * 3) + 1] << "," << vIndex[(currentFace * 3) + 2] << ">"
<< endl;
                currentFace++;
            }
            else
            {
                vIndex[(currentFace * 3) + 0] = i + 1;
                vIndex[(currentFace * 3) + 1] = i;
                vIndex[(currentFace * 3) + 2] = m_numShapeVertices - 2;

                cout << "Face " << currentFace << " <" << vIndex[(currentFace * 3) + 0]
<< "," << vIndex[(currentFace * 3) + 1] << "," << vIndex[(currentFace * 3) + 2] << ">"
<< endl;
                currentFace++;
            }
        }

        else if (i >= stacks && i < m_numShapeVertices - 2) //Orders mid faces
        {
            if (currentSlice == slices - 1) //Last Vertex in Slice
            {
                vIndex[(currentFace * 3) + 0] = i;
                vIndex[(currentFace * 3) + 1] = i - (slices);
                vIndex[(currentFace * 3) + 2] = i - (slices + (slices - 1));

                cout << "Face " << currentFace << " <" << vIndex[(currentFace * 3) + 0]
<< "," << vIndex[(currentFace * 3) + 1] << "," << vIndex[(currentFace * 3) + 2] << ">"
<< endl;
                currentFace++;

                vIndex[(currentFace * 3) + 0] = i;
                vIndex[(currentFace * 3) + 1] = i - 1;
                vIndex[(currentFace * 3) + 2] = i - slices;

                cout << "Face " << currentFace << " <" << vIndex[(currentFace * 3) + 0]
<< "," << vIndex[(currentFace * 3) + 1] << "," << vIndex[(currentFace * 3) + 2] << ">"
<< endl;
                currentFace++;
                currentSlice = 0;
            }
            else if (currentSlice == 0) //First Vertex in Slice
            {
                vIndex[(currentFace * 3) + 0] = i;
                vIndex[(currentFace * 3) + 1] = i - (slices);
                vIndex[(currentFace * 3) + 2] = i - (slices - 1);

                cout << "Face " << currentFace << " <" << vIndex[(currentFace * 3) + 0]
<< "," << vIndex[(currentFace * 3) + 1] << "," << vIndex[(currentFace * 3) + 2] << ">"
<< endl;
                currentFace++;

                vIndex[(currentFace * 3) + 0] = i;
                vIndex[(currentFace * 3) + 1] = i + (slices - 1);
                vIndex[(currentFace * 3) + 2] = i - slices;

                cout << "Face " << currentFace << " <" << vIndex[(currentFace * 3) + 0]
<< "," << vIndex[(currentFace * 3) + 1] << "," << vIndex[(currentFace * 3) + 2] << ">"
<< endl;
                currentFace++;

                currentSlice++;
            }
            else //Vertex Between Slices
            {
                vIndex[(currentFace * 3) + 0] = i;
                vIndex[(currentFace * 3) + 1] = i - (slices);
                vIndex[(currentFace * 3) + 2] = i - (slices - 1);

                cout << "Face " << currentFace << " <" << vIndex[(currentFace * 3) + 0]
<< "," << vIndex[(currentFace * 3) + 1] << "," << vIndex[(currentFace * 3) + 2] << ">"
<< endl;
                currentFace++;

                vIndex[(currentFace * 3) + 0] = i;
                vIndex[(currentFace * 3) + 1] = i - 1;
                vIndex[(currentFace * 3) + 2] = i - slices;

                cout << "Face " << currentFace << " <" << vIndex[(currentFace * 3) + 0]
<< "," << vIndex[(currentFace * 3) + 1] << "," << vIndex[(currentFace * 3) + 2] << ">"
<< endl;
                currentFace++;

                currentSlice++;
            }
        }

        else //Orders bottom faces
        {
            for (int j = (m_numShapeVertices-1) - (slices + 1); j < m_numShapeVertices
- 2; j++)
            {
                if (j < m_numShapeVertices - 3)
                {
                    vIndex[(currentFace * 3) + 0] = m_numShapeVertices - 1;
                    vIndex[(currentFace * 3) + 1] = j;
                    vIndex[(currentFace * 3) + 2] = j+1;

                    cout << "Face " << currentFace << " <" << vIndex[(currentFace * 3) + 0] << "," << vIndex[(currentFace * 3) + 1] << "," << vIndex[(currentFace * 3) + 2] << ">" << endl;
                    currentFace++;
                }
                else
                {
                    vIndex[(currentFace * 3) + 0] = m_numShapeVertices - 1;
                    vIndex[(currentFace * 3) + 1] = j;
                    vIndex[(currentFace * 3) + 2] = j- (slices - 1);

                    cout << "Face " << currentFace << " <" << vIndex[(currentFace * 3) + 0] << "," << vIndex[(currentFace * 3) + 1] << "," << vIndex[(currentFace * 3) + 2] << ">" << endl;
                    currentFace++;
                }
            }
            i = m_numShapeVertices;
        }
        //phi2  phi1
        // |     |
        // 2-----1 -- theta1
        // | \   |
        // |   \ |
        // 3-----4 -- theta2
    }
```