| LCOV - code coverage report |
| --- |
|  |
| | Current view: | [top level](http://docs.google.com/index.html) - [Common/include](http://docs.google.com/index.html) - Rect.h (source / [functions](http://docs.google.com/Rect.h.func.html)) |  |  | Hit | Total | Coverage | | --- | --- | --- | --- | --- | --- | --- | | Test: | EBM9K Converter Coverage Report |  | Lines: | 41 | 42 | 97.6 % | | Date: | 2012-03-23 |  | Functions: | 18 | 18 | 100.0 % | | Legend: | Lines: hit not hit | Branches: + taken - not taken # not executed |  | Branches: | 5 | 10 | 50.0 % | |  | |
|  |

|  |
| --- |
| Branch data Line data Source code  1 : : #ifndef \_RECT\_H\_  2 : : #define \_RECT\_H\_  3 : :   4 : : #include <stdint.h>  5 : : #include <algorithm>  6 : :   7 : : #include "Position.h"  8 : : #include "Memory.h"  9 : :   10 : : namespace nft {  11 : :   12 : : template <typename \_Tp>  13 : : class Rect : public NftNewDeleteBase {  14 : : public:  15 : : Rect();  16 : :   17 : : Rect(const Position<\_Tp> &llCorner, const Position<\_Tp> &size);  18 : :   19 : : ~Rect();  20 : :   21 : : void  22 : : setLowerLeftCorner(const Position<\_Tp> &value);  23 : :   24 : : void  25 : : setSize(const Position<\_Tp> &value);  26 : :   27 : : inline const Position<\_Tp>  28 : : getLowerLeftCorner() const;  29 : :   30 : : inline const Position<\_Tp>  31 : : getSize() const;   32 : :   33 : : const Position<\_Tp>  34 : : getUR() const; //get upper right corner   35 : :   36 : : Rect(const Rect<\_Tp> & source);  37 : :   38 : : Rect<\_Tp> &  39 : : operator = (const Rect<\_Tp> & source);  40 : :   41 : : bool  42 : : operator == (const Rect<\_Tp> & rhs) const;  43 : :   44 : : bool  45 : : checkIsIncluded(const Rect<\_Tp> &inValue) const;  46 : :   47 : : Rect<\_Tp>  48 : : rectUnion(const Rect<\_Tp> & rhs);  49 : :   50 : : Rect<\_Tp>  51 : : rectIntersection(const Rect<\_Tp> & rhs);  52 : : bool  53 : : checkIsPointIncluded(const Position<\_Tp> &p) const;  54 : :   55 : : private:  56 : :   57 : : Position<\_Tp> mLowerLeftCorner;   58 : :   59 : : Position<\_Tp> mSize;  60 : : };  61 : :   62 : : typedef Rect<uint64\_t> ULongRect;  63 : : typedef Rect<int64\_t> LongRect;  64 : : typedef Rect<uint32\_t> UIntRect;  65 : : typedef Rect<int32\_t> IntRect;  66 : : typedef Rect<uint16\_t> UShortRect;  67 : : typedef Rect<int16\_t> ShortRect;  68 : :   69 : : template <typename \_Tp>  70 : : Rect<\_Tp>::Rect()  71 : 3756 : :mLowerLeftCorner(),  72 : 3756 : mSize() {   73 : : }  74 : :   75 : : template <typename \_Tp>  76 : : Rect<\_Tp>::Rect(const Position<\_Tp> &llCorner, const Position<\_Tp> &size)  77 : 4659 : :mLowerLeftCorner(llCorner),  78 : 4659 : mSize(size) {  79 : : }  80 : :   81 : : template <typename \_Tp>  82 : 8415 : Rect<\_Tp>::~Rect() {  83 : : }  84 : :   85 : : template <typename \_Tp>  86 : : void  87 : 2258 : Rect<\_Tp>::setLowerLeftCorner(const Position<\_Tp> &value) {  88 : 2258 : mLowerLeftCorner = value;   89 : : }  90 : :   91 : : template <typename \_Tp>  92 : : void  93 : 2258 : Rect<\_Tp>::setSize(const Position<\_Tp> &value) {  94 : 2258 : mSize = value;  95 : : }  96 : :   97 : : template <typename \_Tp>  98 : : inline const Position<\_Tp>   99 : 2980 : Rect<\_Tp>::getLowerLeftCorner() const {  100 : : return mLowerLeftCorner;   101 : : }  102 : :   103 : : template <typename \_Tp>  104 : : inline const Position<\_Tp>  105 : 1585 : Rect<\_Tp>::getSize() const {  106 : : return mSize;  107 : : }  108 : :   109 : : template <typename \_Tp>  110 : : const Position<\_Tp>  111 : 352 : Rect<\_Tp>::getUR() const {  112 : 352 : return Position<\_Tp>(mLowerLeftCorner.getX() + mSize.getX(),  113 : : mLowerLeftCorner.getY() + mSize.getY());  114 : : }  115 : :   116 : : template <typename \_Tp>  117 : : Rect<\_Tp>::Rect(const Rect<\_Tp> & source)   118 : : :mLowerLeftCorner(source.mLowerLeftCorner),  119 : : mSize(source.mSize) {  120 : : }  121 : :   122 : : template <typename \_Tp>  123 : : Rect<\_Tp> &  124 : 1342 : Rect<\_Tp>::operator = (const Rect<\_Tp> & source) {  125 [ + - ]: 1342 : if(this != &source) {  126 : 1342 : mLowerLeftCorner = source.mLowerLeftCorner;  127 : 1342 : mSize = source.mSize;  128 : : }  129 : 1342 : return(\*this);  130 : : }  131 : :   132 : : template <typename \_Tp>  133 : : bool  134 : 232 : Rect<\_Tp>::operator == (const Rect<\_Tp> & rhs) const {  135 [ + - ][ + - ]: 232 : return(mLowerLeftCorner == rhs.mLowerLeftCorner && mSize == rhs.mSize);  136 : : }  137 : :   138 : : template <typename \_Tp>  139 : : bool  140 : 324 : Rect<\_Tp>::checkIsPointIncluded(const Position<\_Tp> &p) const {  141 : 324 : if( p.getX() >= mLowerLeftCorner.getX()  142 : : && p.getX() < (mLowerLeftCorner.getX() + mSize.getX())) {  143 : 324 : if( p.getY() >= mLowerLeftCorner.getY()  144 : : && p.getY() < (mLowerLeftCorner.getY() + mSize.getY())) {  145 : 324 : return true;  146 : : }  147 : : }  148 : 324 : return false;  149 : : }  150 : :   151 : : template <typename \_Tp>  152 : : bool  153 : 16 : Rect<\_Tp>::checkIsIncluded(const Rect<\_Tp> &inValue) const {  154 : : Position<\_Tp> URCorner = Position<\_Tp>(  155 : : inValue.mLowerLeftCorner.getX() + inValue.mSize.getX(),   156 : 16 : inValue.mLowerLeftCorner.getY() + inValue.mSize.getY());   157 : : //check if LL corner & UR corner is insdie rectangle  158 [ + - ][ + - ]: 16 : return (checkIsPointIncluded(inValue.mLowerLeftCorner)) &&   159 : : (checkIsPointIncluded(URCorner));  160 : :   161 : : }  162 : : template <typename \_Tp>  163 : : Rect<\_Tp>  164 : : Rect<\_Tp>::rectUnion(const Rect<\_Tp> & rhs) {  165 : : \_Tp LLxMin = std::min(mLowerLeftCorner.getX(), rhs.mLowerLeftCorner.getX());  166 : : \_Tp LLyMin = std::min(mLowerLeftCorner.getY(), rhs.mLowerLeftCorner.getY());  167 : : \_Tp rightCornerXMax = std::max(mLowerLeftCorner.getX() + mSize.getX(),  168 : : rhs.mLowerLeftCorner.getX() + rhs.mSize.getX());   169 : : \_Tp rightCornerYMax = std::max(mLowerLeftCorner.getY() + mSize.getY(),  170 : : rhs.mLowerLeftCorner.getY() + rhs.mSize.getY());  171 : : \_Tp sizeX = rightCornerXMax - LLxMin;  172 : : \_Tp sizeY = rightCornerYMax - LLyMin;  173 : : return Rect<\_Tp>(Position<\_Tp>(LLxMin,LLyMin), Position<\_Tp>(sizeX, sizeY));   174 : : }  175 : :   176 : : template <typename \_Tp>  177 : : Rect<\_Tp>  178 : 1164 : Rect<\_Tp>::rectIntersection(const Rect<\_Tp> & rhs) {  179 : 1164 : \_Tp rightXofRect1 = mLowerLeftCorner.getX() + mSize.getX();  180 : 1164 : \_Tp rightYofRect1 = mLowerLeftCorner.getY() + mSize.getY();  181 : 1164 : \_Tp rightXofRect2 = rhs.mLowerLeftCorner.getX() + rhs.mSize.getX();  182 : 1164 : \_Tp rightYofRect2 = rhs.mLowerLeftCorner.getY() + rhs.mSize.getY();  183 : 1164 : if((mLowerLeftCorner.getX() >= rightXofRect2) ||  184 : : (mLowerLeftCorner.getY() >= rightYofRect2) ||  185 : : (rhs.mLowerLeftCorner.getX() >= rightXofRect1) ||  186 : : (rhs.mLowerLeftCorner.getY() >= rightYofRect1)) {  187 : 0 : return Rect<\_Tp>(Position<\_Tp>(0,0), Position<\_Tp>(0,0));  188 : : } else {  189 : : \_Tp LLxMax = std::max(mLowerLeftCorner.getX(),  190 : 1164 : rhs.mLowerLeftCorner.getX());  191 : : \_Tp LLyMax = std::max(mLowerLeftCorner.getY(),  192 : 1164 : rhs.mLowerLeftCorner.getY());  193 : 1164 : \_Tp RxMin = std::min(rightXofRect1, rightXofRect2);  194 : 1164 : \_Tp RyMin = std::min(rightYofRect1, rightYofRect2);  195 : 1164 : \_Tp sizeX = RxMin - LLxMax;  196 : 1164 : \_Tp sizeY = RyMin - LLyMax;  197 : 1164 : return Rect<\_Tp>(Position<\_Tp>(LLxMax,LLyMax),Position<\_Tp>(sizeX, sizeY));  198 : : }  199 : : }  200 : :   201 : : } //namespace nft  202 : : #endif |

|  |
| --- |
| Generated by: [LCOV version 1.9](http://ltp.sourceforge.net/coverage/lcov.php) |