■今日の稽古

中心をとるために，意識して左の方に踏み込むように練習.
それなりに中心をとることができたかな.
ただ意識しすぎて体が硬くなっているのか,
相手のすばやい動きに追従するのが若干困難.
あとは，左足の遊び癖もなおさないと.

■ COINS での構造体

coins/hir2lir/ConvToNewLir 1854

```
          case HIR.OP_ARROW:
            return
              convertLoadMember
              (convertNode(((PointedExp)pHir).getPointerExp(), cc),
               ((PointedExp)pHir).getPointedElem(),
               type, cc);
```

coins/hir2lir/ConvToNewLir 2909

```
       /** Convert expression loading x.member / x->member. **/
       private ImList convertLoadMember(ImList base, Elem elem,
                                        Type type, ConvContext cc) {
         String ltype = htype2ltype(type);
         String inttype = hkind2ltype(Type.KIND_INT);
         String ptrtype = hkind2ltype(Type.KIND_POINTER);

         long offset = elem.evaluateDisp();
```

coins/sym/ElemImpl 89

```
      public long evaluateDisp()
      {
       if (fDispIsSet)
         return fDisplacement;
       if (fDispExp != null)  {
         fDisplacement = fDispExp.evaluateAsInt();
         fDispIsSet    = true;
         return fDisplacement;
       }else {
         symRoot.ioRoot.msgRecovered.put(1014, "Displacement is not evaluable "
                                 + getName());
         return 0;
       }
      } // evaluateDisp
```

coins/sym/ElemImpl 109

```
      public void
      setDisplacement( long pDisplacement ) {
       fDisplacement = pDisplacement;
       fDispIsSet    = true;
      }
```

こういう構造体を与えた時

```
      struct bitMapFileHeader{
        unsigned short bfType __attribute__((packed));
```

```
  unsigned int bfSize __attribute__((packed)); // 2
  unsigned short bfReserved1 __attribute__((packed)); // 6
  unsigned short bfReserved2 __attribute__((packed)); // 8
  unsigned int offBits __attribute__((packed)); // 10
};
```

setDisplacement の後の ElemImpl のダンプ結果

```
bfType 0 disp 0
bfSize 0 disp 4
bfReserved1 0 disp 8
bfReserved2 0 disp 10
offBits 0 disp 12
```

この setDisplacement の呼出元は

coins/sym/StructImple 223

```
      }else { // Not a bit field.
        if (lBitFieldBegin) { // Previous fields are bit field.
          lSize = lContainingObjectDisplacement
                + (lBitFieldSumWithinWord+7) / 8;
          lBitFieldSumWithinWord = 0;
        }
        lBitFieldBegin = false;
        lDisplacement = lSize + lElemType.getAlignmentGap(lSize);
        lContainingObjectDisplacement = lDisplacement;
        lElem.setDisplacement(lDisplacement);
        //## lElem.setDispExp( ... );   // REFINE
        lSize = lDisplacement + lElemSize;
        lElemAlignment = lElemType.getAlignment();
        if (lElemType.getFlag(Sym.FLAG_INCOMPLETE_TYPE)) {
          lIncomplete = true;
          lUniformSize = false;
        }
        if (! lElemType.getFlag(Sym.FLAG_UNIFORM_SIZE))
          lUniformSize = false;
        if (! lElemType.isSizeEvaluable())
          lSizeIsSet = false;
      }
```

この中の lElemType は TypeImpl を継承したクラスで

coins/sym/TypeImpl 468

```
public int                              // to be refined
getAlignmentGap( long pPrecedingSize ) {
 int lAlignment, lAlignmentGap;
 long lResidue;
 lAlignment = getAlignment();
 lResidue = pPrecedingSize % lAlignment;
 if (lResidue == 0)
   lAlignmentGap = 0;
 else
   lAlignmentGap = (int)(lAlignment - lResidue);
 return lAlignmentGap;
} // getAlignmentGap
```

coins/sym/TypeImpl 468

```
public int
getAlignment()
{
 int lAlignment;
```

```
 //##67 if (fTypeKind == Type.KIND_SUBP) {
 //##67   lAlignment = ((SubpType)this).getReturnType().getAlignment();
 //##67 }else {
 lAlignment = Type.KIND_ALIGNMENT[fTypeKind];
            // STRUCT, UNION, VECTOR have their own method.
 //##67 }
 return lAlignment;
}
```

coins/sym/TypeImple 550

public void

```
setStaticTable(MachineParam pMachineParam)
{
 for (int i = 0; i < KIND_ALIGNMENT.length; i++) {    //##52
   KIND_ALIGNMENT[i] = pMachineParam.getAlignment(i); //##52
 }                                                    //##52
 /* //##52
```

coins/sym/SymRoot 304

```
 ioRoot.symRoot = this; //##12
 ((SymImpl)sym).setParameters(machineParam, sourceLanguage); //##51
 (new TypeImpl(this)).setStaticTable(machineParam);          //##51
} // SymRoot
```

machineParam は、coins/MachinParamXXX にある。デフォルトの場合は coins/Machinparam