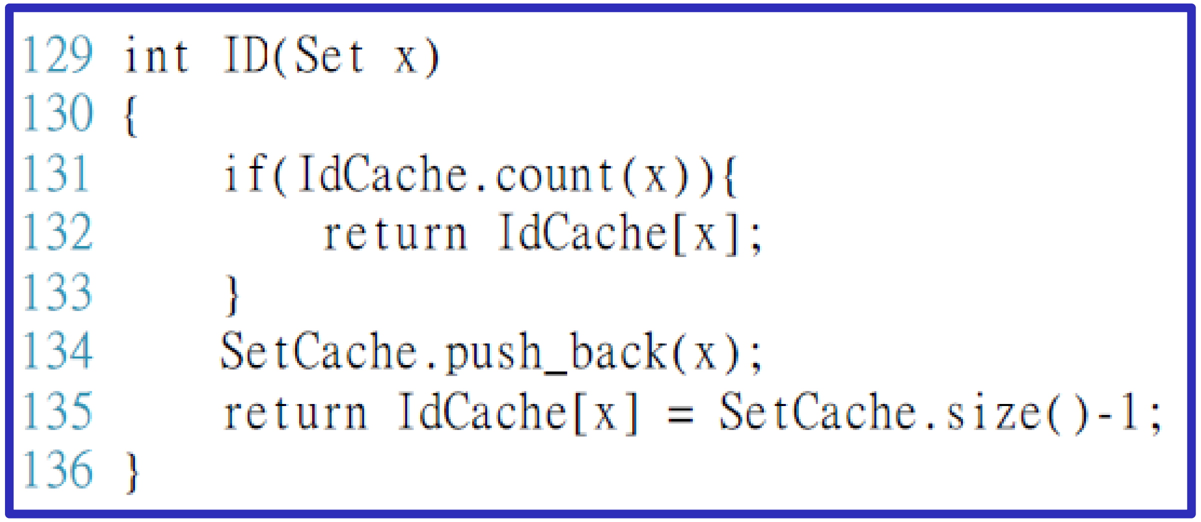
計算機韌體實驗（P17）集合堆疊電腦/The SetStack Computer

饒建奇（Jiann-Chyi Rau）
淡江大學電機工程學系，E627
E-mail: jcrau@ee.tku.edu.tw

解題要訣

- 尋找給定集合(set) x的ID，如果找不到，分配一個新的ID
  - 利用映射(map)來記錄x和其對應的ID
  - 利用x的ID來操作堆疊(stack)運算

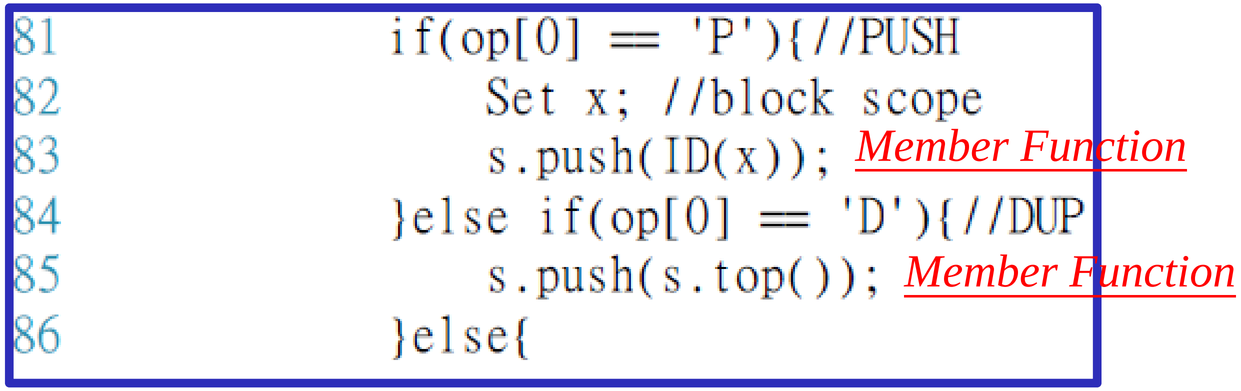


```
int ID(Set x)
{
    if(IdCache.count(x)){
        return IdCache[x];
    }
    SetCache.push_back(x);
    return IdCache[x] = SetCache.size()-1;
}
```

堆疊: stack

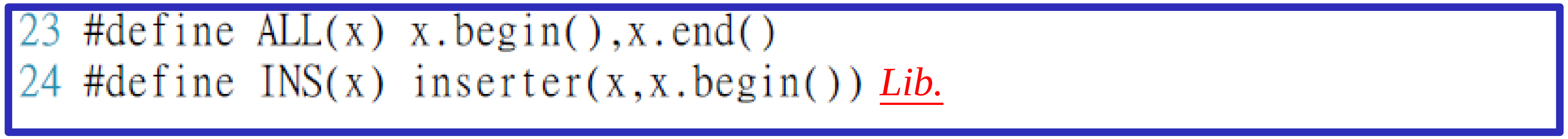
- stack<*type*>是一種先進後出(First-In Last-Out, FILO)的容器
  - stack is define in <stack>
- 主要運算皆發生於頂端(top)
  - 推入(push)
  - 彈出(pop)

堆疊運算: PUSH和DUP

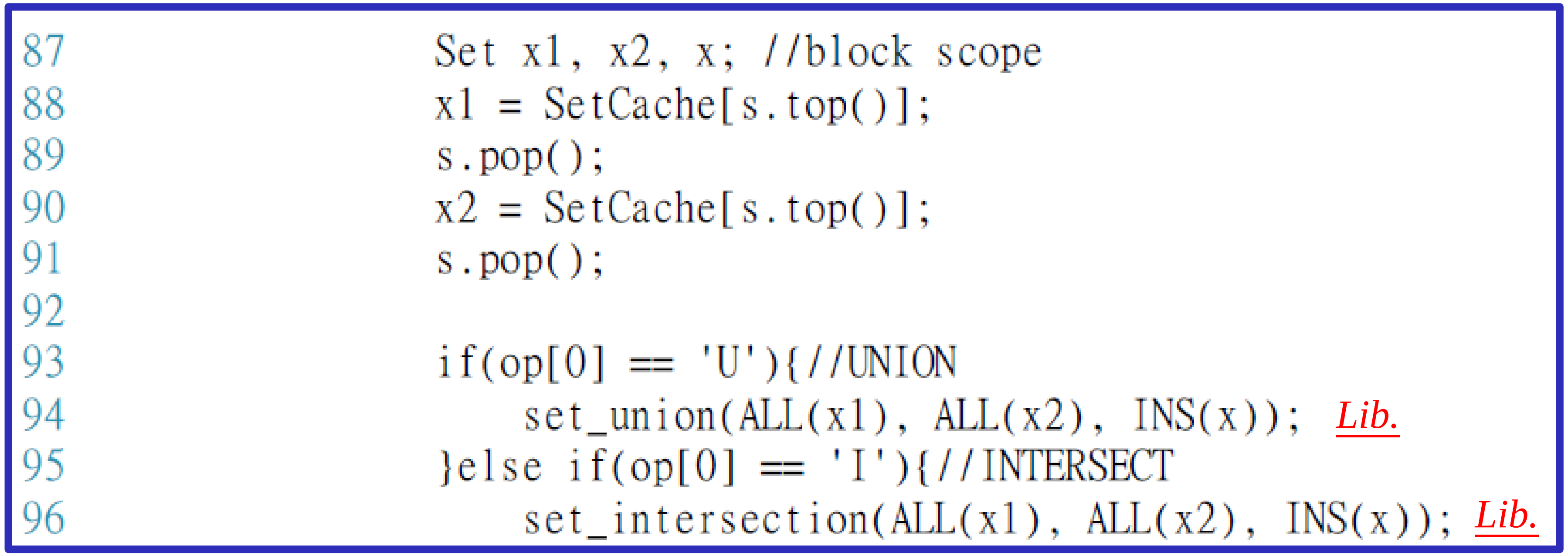


```
            if(op[0] == 'P'){//PUSH
                Set x; //block scope
                s.push(ID(x)); Member Function
            }else if(op[0] == 'D'){//DUP
                s.push(s.top()); Member Function
            }else{
```

堆疊運算: UNION和INTERSECT



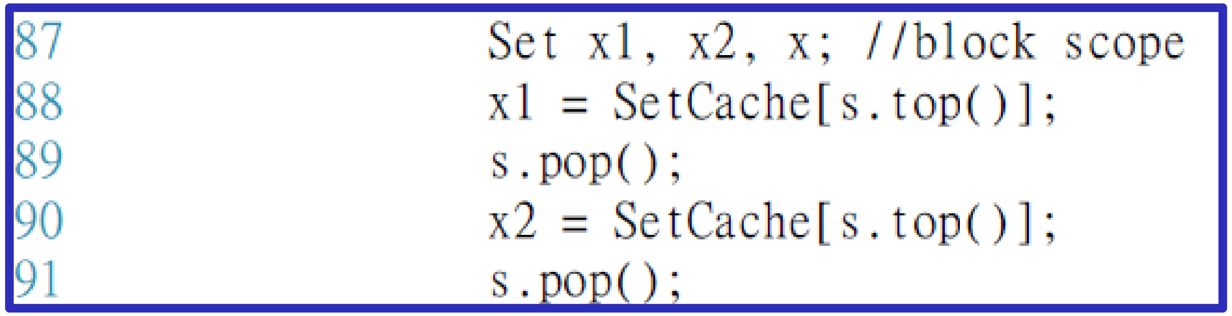
```
#define ALL(x) x.begin(),x.end()
#define INS(x) inserter(x,x.begin()) Lib.
```



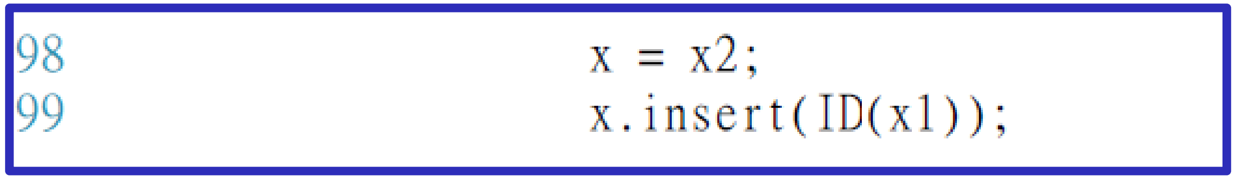
```
Set x1, x2, x; //block scope
x1 = SetCache[s.top()];
s.pop();
x2 = SetCache[s.top()];
s.pop();

if(op[0] == 'U'){//UNION
    set_union(ALL(x1), ALL(x2), INS(x));  Lib.
}else if(op[0] == 'I'){//INTERSECT
    set_intersection(ALL(x1), ALL(x2), INS(x)); Lib.
```

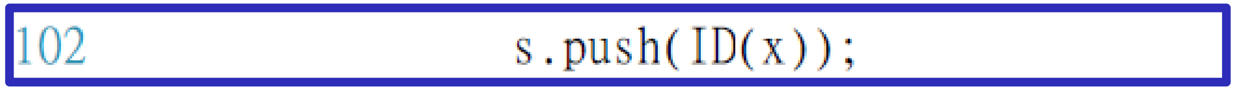
堆疊運算: ADD



```
                Set x1, x2, x; //block scope
                x1 = SetCache[s.top()];
                s.pop();
                x2 = SetCache[s.top()];
                s.pop();
```



```
                    x = x2;
                    x.insert(ID(x1));
```



```
                  s.push(ID(x));
```

Lib. Functions

- inserter(x, x.begin())
  - Defined in <iterator>
  - Return an insert iterator for x (*x's iterator*)
- set_union(first1, last1, first2, last2, *x's iterator*)
  - Define in <algorithm>
  - Let x = [first1, last1] ∪ [first2, last2]
- set_intersection(first1, last1, first2, last2, *x's iterator*)
  - Define in <algorithm>
  - Let x = [first1, last1] ∩ [first2, last2]

Member Functions

- void stack::push(const value_type &val);
  - Insert a copy of val at the top of the stack, i.e., above its current top element
- value_type statck::top(void);
  - Return a reference to the top element in the stack
- void stack::pop(void)
  - Remove the element on the top of the stack, effectively reducing its size by one