# 第一次报告

物1802孙肖扬20184425

这几天我负责的是符号表相关设计，从原先的艰难，一点都没有天绪到现在内心的欣喜;从之前连编译都不能通过，到现在的初步入门，中间经历了很多很多。从中我学习了很多，了解了很多，知道了很多。首先对实验原理有更深的理解  
通过编译原理课程设计，掌握了什么是编译程序，编译程序工作的基本过程及其各阶段的基本任务，熟悉了编译程序总流程框图，了 解了编译程序的生成过程、构造工具及其相关的技术对课本上的知识有了更深的理解，课本上的知识师机械的，表面的。通过把该算法的内容，算法的执行顺序在计算机上实现，把原来以为很深奥的书本知识变的更为简单，对实验原理有更深的理解，对该理论在实践中的应用有深刻的理解，通过把该算法的内容，算法的执行顺序在计算机上实现，知道和理解了该理论在计算机中是怎样执行的，对该理论在实践中的应用有深刻的理解。

符号表结构：

#include<stdio.h>

#include<string.h>

#include<ctype.h>

#include<stdlib.h>

Struct

{  
int m;  
char name 【20】；  
char inf[20」；}

co[999];  
int num;

//预定义  
struct snode;  
struct stable;  
//符号表结点  
struct snode  
{  
 string text;  //符号名称  
 string type;  //符号类型  
 union {int ival;double rval;}value;  //值------------  
 int offset;   //偏移量  
 snode \*nextn;  //指向下一个节点  
 stable \*header;  //指向下属符号表的表头  
};  
//符号表表头  
struct stable  
{  
 stable \*previous; //指向先前创建的符号表表头  
 snode \*firstnode; //指向第一个结点  
 stable \*ifnoelements;//如果此表为空，则用它指向下一个表头  
};  
   
//当前表头  
stable \*currtab;  
//建立新表，返回表头指针  
//参数：当前的节点的表头  
stable \*mktable(stable \*previous)  
{  
 stable \*newtable =new stable;  
 newtable->previous=previous;  
 newtable->ifnoelements=0;  
 newtable->firstnode=0;  
 if(previous->firstnode==0)  
 {  
  previous->ifnoelements=newtable;  
 }  
 else  
 {  
  snode\* ininode=previous->firstnode;  
  while(ininode->nextn!=0)  
  {  
   ininode=ininode->nextn;  
  }  
  ininode->header=newtable;  
 }  
   
 currtab=newtable;  
 return newtable;  
}  
//在node指向的符号表中为text建立一个新表项，返回新建立的结点  
//参数：node为当前的节点的表头，text名称，type类型，offset偏移  
snode \*enter(stable \*table,string text,string type,int offset,double value)  
{  
   
    
  //创建节点  
  snode \*newnode = new snode;  
  newnode->text=text;  
  newnode->type=type;  
  newnode->offset=offset;  
  newnode->nextn=0;  
  newnode->header=0;  
  if(type=="int")  
  {  
   newnode->value.ival=value;  
  }  
  else if(type=="real")  
  {  
   newnode->value.rval=value;  
  }  
    
  //判断此表是否无元素  
  if(currtab->firstnode==0)  
  {  
   currtab->firstnode=newnode;  
   currtab->ifnoelements=0;  
  }  
  else  
  {  
   snode\* addnode=currtab->firstnode;  
   while(addnode->nextn!=0)  
   {  
    addnode=addnode->nextn;  
   }  
   addnode->nextn=newnode;  
  }  
    
  return newnode;  
}  
//初始化符号表，返回表头节点  
void inittab()  
{  
 stable \*initable = new stable;  
 initable->firstnode=0;  
 initable->previous=0;  
 initable->ifnoelements=0;  
 currtab=initable;  
}  
//查找指针，表示结果  
snode \*searchresult;  
//查找变量，返回指向该变量的指针  
//查找变量，返回指向该变量的指针  
snode\* search(string name)  
{  
 //检查表是否空  
 bool isempty=true;  
 stable\* checktab=currtab;  
 if(checktab->firstnode!=0)  
 {isempty=false;}  
 while(checktab->previous!=0)  
 {  
  if(checktab->firstnode!=0)  
  {isempty=false;}  
  checktab=checktab->previous;  
 }  
 if(checktab->firstnode!=0)  
 {isempty=false;}  
 if(isempty)  
 {  
  return 0;  
 }  
 snode\* lastnode;  
 if(currtab->firstnode==0)  
 {  
  //移到非空的表头  
  stable\* notnullhead=currtab;  
  while(notnullhead->firstnode==0)  
  {  
   notnullhead=notnullhead->previous;  
  }  
  snode\* tmpnode=notnullhead->firstnode;  
  //移到最后的元素  
  while(tmpnode->nextn!=0)  
  {  
   tmpnode=tmpnode->nextn;  
  }  
  lastnode=tmpnode;  
 }  
 else  
 {  
  lastnode=currtab->firstnode;  
  while(lastnode->nextn!=0)  
  {  
   lastnode=lastnode->nextn;  
  }  
 }  
 //移到表头  
 stable\* fronttab=currtab;  
 while(fronttab->previous!=0)  
 {  
  fronttab=fronttab->previous;  
 }  
 snode\* nownode=0;  
 while(nownode!=lastnode)  
 {  
  while(fronttab->ifnoelements!=0)  
  {  
   fronttab=fronttab->ifnoelements;  
  }  
  nownode=fronttab->firstnode;  
  while(nownode->nextn!=0)  
  {  
   if(nownode->text==name)  
   {  
    searchresult=nownode;  
    return searchresult;  
   }  
   nownode=nownode->nextn;  
  }  
  if(nownode->text==name)  
  {  
   searchresult=nownode;  
   return searchresult;  
  }  
  fronttab=nownode->header;  
 }  
 if(nownode->text==name)  
 {  
  searchresult=nownode;  
  return searchresult;  
 }  
return 0;  
}  
   
//消毁符号表  
void delNode()  
{  
//more codes here......  
}