КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Лабораторна робота №3

з дисципліни «Системне програмування»

Залікова книжка № 4213

Виконав студент 3-го курсу
групи ІО-42
Кочетов Данило

2016

Мета: Вивчення схеми табличного подання автоматної граматики лексичного аналізу. Використання об'єктів стану графів автоматів для формування лексем у форматі внутрішнього подання вузлів графів розбору.

№ вар.	Вираз, який відтворюється в графі внутрішнього подання	Мова відтворення
13	**for**(b=0;n;n--)b+=a[n];	C

Лістинг програми

lab3.cpp

```
#include "stdafx.h"
#include "..\..\lab2\lab2\token.h"
#include "..\..\lab2\lab2\visgrp.h"
#include "tables.h"
#include "lexan.h"
#include "langio.h"
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

extern struct recrdKWD *tablKWD;
struct lxNode nodes[200]= // масив приймач вузлів дерева
{{_nil,NULL,NULL,0,0,0,0,0,NULL,0},{_nil,NULL,NULL,0,0,0,0,0,NULL,0},
{_nil,NULL,NULL,0,0,0,0,0,NULL,0},{_nil,NULL,NULL,0,0,0,0,0,NULL,0},
{_nil,NULL,NULL,0,0,0,0,0,NULL,0},{_nil,NULL,NULL,0,0,0,0,0,NULL,0},
{_nil,NULL,NULL,0,0,0,0,0,NULL,0},{_nil,NULL,NULL,0,0,0,0,0,NULL,0},
{_nil,NULL,NULL,0,0,0,0,0,NULL,0},{_nil,NULL,NULL,0,0,0,0,0,NULL,0},
{_nil,NULL,NULL,0,0,0,0,0,NULL,0},{_nil,NULL,NULL,0,0,0,0,0,NULL,0},
{_nil,NULL,NULL,0,0,0,0,0,NULL,0},{_nil,NULL,NULL,0,0,0,0,0,NULL,0},
{_nil,NULL,NULL,0,0,0,0,0,NULL,0},{_nil,NULL,NULL,0,0,0,0,0,NULL,0},
{_nil,NULL,NULL,0,0,0,0,0,NULL,0},{_nil,NULL,NULL,0,0,0,0,0,NULL,0},
{_nil,NULL,NULL,0,0,0,0,0,NULL,0},{_nil,NULL,NULL,0,0,0,0,0,NULL,0},
};
char file_name[20];
extern enum ltrType ltClsC[256];
extern enum ltrType ltClsP[256];
extern enum tokType dlCdsC[256];
extern enum tokType dlCdsP[256];
extern enum ltrType ltClsC[256];
extern enum ltrType ltClsP[256];
enum ltrType *ltCls=ltClsC;
enum tokType *dlCds=dlCdsC;
int main(int argc, char* argv[])
{int nn=-1;//np,
 if (argc>1)
    {strcpy(file_name,argv[1]);
        printf("Processing file -- %s\n",file_name);}
 else
    {printf("Please enter file Name: ");
        scanf("%s",file_name);
        strcat(file_name,".h");
       }
 opFls(file_name);
 LxAnInit('C');
 srtBin(tablKWD, 67);
do{//np=nn;
 nn=LxAnlzr();
}while(nodes[nn].ndOp!=_EOF);
      prLaTxt(nodes,nn);
      printf("\n");
 return 0;
}
```

Index.cpp

```
#include "stdafx.h"
#include "..\..\lab2\lab2\token.h"
#include "tables.h"
#include "index.h"
#include <stdlib.h>
// порівняння рядків
// порівняння терміналів за відношенням порядку
int cmpTrm(struct lxNode*k0,struct lxNode*kArg)//cmpKys
{int i=cmpStr((unsigned char*)k0->prvNd,
                      (unsigned char*)kArg->prvNd);
 if(i)return i;
 return k0->stkLength - kArg->stkLength; // порівняння номерів модулів
}
unsigned nNdxNds=0;
struct indStrUS ndxNds[50]={{NULL,NULL,NULL,0}},
             *pRtNdx=ndxNds, nilNds={NULL,NULL,NULL,0};
// вибірка через пошук за двійковим деревом
struct indStrUS*selBTr(struct lxNode*kArg,struct indStrUS*rtTb)
{int df;
 while(df=cmpTrm(kArg,rtTb->pKyStr))
       if(df>0){if(rtTb->pRtPtr)rtTb=rtTb->pRtPtr;
                      else break;}
       else{if(rtTb->pLtPtr)rtTb=rtTb->pLtPtr;
              else break;}
 rtTb->dif=df;
 return rtTb;
}
// включення через пошук за двійковим деревом
struct indStrUS*insBTr(struct lxNode*pElm,struct indStrUS*rtTb)
{struct indStrUS*pInsNod;//,*pNod;
 if(rtTb->pKyStr==NULL)
 {rtTb->pKyStr=pElm;
  return rtTb;
 }
// if(rtTb->pKyStr->ndOp==_nil)rtTb->pKyStr=pElm;
 else{pInsNod=selBTr(pElm,rtTb);
        if(pInsNod->dif)
        {ndxNds[++nNdxNds]=nilNds;
         if(pInsNod->dif<0)pInsNod=pInsNod->pLtPtr=ndxNds+nNdxNds;
         else pInsNod=pInsNod->pRtPtr=ndxNds+nNdxNds;
         ndxNds[nNdxNds].pKyStr=pElm;
        }
       }
 return pInsNod;
}
```

Lexan.cpp

```
#include "stdafx.h"
#include "automat.h"
#include "langio.h"
#include "..\..\lab2\lab2\token.h"
#include "tables.h"
#include "index.h"
#include <stdlib.h>
extern enum ltrType *ltCls;// уточнюються для версій та режимів
extern enum autStat nxtStsC[Eo+1][ltrcode+1];
extern enum autStat nxtStsP[Eo+1][ltrcode+1];
extern char *oprtrC[], *oprtrP[], *oprtrV[],
                    *cprC[], *cprP[], *cprV[];
char **oprtr, **cpr;
char modeP=0, // тип роздільника операторних дужок для Паскаля
        modeC=1, // тип роздільника операторних дужок для С
        modeL;
extern int x, y, f;
extern unsigned     nImBg, nImCr; // Початковий та поточний номери образів в буфері
extern char imgBuf[];      // буфер вхідних образів
extern enum tokType *dlCds;
extern enum ltrType *ltCls;
```

```
unsigned nNode=0;
extern struct lxNode nodes[100];
extern unsigned nNdxNds;
extern struct indStrUS *pRtNdx, ndxNds[50];
struct recrdKWD *tablKWD;
extern struct recrdKWD tablKWDC[67];
extern struct recrdKWD tablKWDP[67];
enum autStat //*nxtStsR=&nxtStsC[0][0],
                    nxtSts[Eo+1][ltrcode+1];//=&nxtStsR;//(enum autStat**)nxtStsC;
void LxAnInit(char nl)
{char i=0;
 switch (nl)
 {case 'P':modeL=modeP;
        oprtr=oprtrP, cpr=cprP;
        tablKWD = tablKWDP;
        for(i=0;i<=Eo;i++)for(char j=0;j<=ltrcode;j++)
              nxtSts[i][j]=nxtStsP[i][j];
        break;
 case 'V':
 default:
 case 'C':modeL=modeC;
        oprtr=oprtrC, cpr=cprC;
        tablKWD = tablKWDC;
        for(i=0;i<=Eo;i++)for(char j=0;j<=ltrcode;j++)
              nxtSts[i][j]=nxtStsC[i][j];
}
}
// функція лексичного аналізу чергової лексеми
int cntMdB=0;
int LxAnlzr(void)
{//static int lxNmb=0;
 static enum autStat s=S0, sP; // поточний та попередній стан лексеми
 enum autStat SP;
 static enum ltrType c;    // клас чергової літери
 static char l=1;          // чергова літера (початок фыктивний)
 struct recrdKWD* pRt;
 int s1, c1;
 char l1, l0;       // чергова літера
 lxInit(nNode,c);
 do {SP=
        sP=s; l1=l;  // запам'ятовування стану
        l0=l=ReadLtr();     // читання літери
        c1=c=ltCls[l];      // визначення класу літери
        if(s==Scl&&c!=dlmeorml)continue;
        s=nxtSts[s][c];//[c<dlmaux?c:dlmaux];// стан лексеми
        s1=s;
        if(s==Scr)continue;
        if((sP==S2||sP==S3)&&(c>nc&&c<dlmeormr))
        {// пошук в таблиці групових роздільників
              imgBuf[nImCr]=0;
         pRt=selBin(imgBuf+nImBg, tablKWD, 67);
         if(pRt!=NULL)
             {nodes[nNode].ndOp=pRt->func;
              if(pRt->func==_remL)
                    {s=Scr; nodes[nNode].pstNd=(struct lxNode *)(imgBuf+nImCr);
                    continue;}
              if(pRt->func==_rem)
                    {s=Scl;} continue;
              break;}else
              {if(sP!=S3)
              {nodes[nNode].ndOp=/*(enum tokType)*/dlCds[l1];
               imgBuf[nImBg]=imgBuf[nImBg+1];
              }else{imgBuf[nImBg]=imgBuf[nImBg+1]; nImCr--;}
               nImCr--;
               sP=S0;
               s=nxtSts[sP][c];//[c<dlmaux?c:dlmaux];// стан лексеми
               return nNode++;
              }
        }
    s1=s;}while(s!=S0&&s!=S2&&!((sP==S0||sP==S2||sP==S3)&&s<S2));   // перевірка кінця лексеми
```

```
        s1=sP;
  switch(sP)
  {case Scr: case Scl:
        imgBuf[nImCr++]=0;
//      ((char*)(nodes[nNode].prvNd))--;
        nImBg=nImCr;
        break;
   case S2:
   case S0:
/*      if(s==S0)
//            dGroup(nNode);// аналіз групових роздільників
              {//imgBuf[nImBg]=l1;
              imgBuf[++nImCr]=0;//nImCr++;
         }*/
        if(nodes[nNode].ndOp!=_nil)
              {nImCr=nImBg; l=' '; return nNode++;}
//      if(sP!=S0)
        nodes[nNode].ndOp=/*(enum tokType)*/dlCds[l1];//dlCds[l1];
        if(nodes[nNode].ndOp==_opbr&&(nodes[nNode-1].ndOp==_ass||cntMdB))
              {cntMdB++;nodes[nNode].ndOp=_tdbr;}
        if(nodes[nNode].ndOp==_ocbr&&cntMdB)
              {nodes[nNode].ndOp=_tcbr;cntMdB--;}
        if(nodes[nNode].ndOp!=_nil)
//            &&imgBuf[nImBg]==)
              {nodes[nNode].prvNd=NULL;
               if(nImBg+1!=nImCr)
//             if(dlCdsC[l0]!=_nil||ltClsC[l0]==dlmaux||ltClsC[l0]==dlmeormr)
               {imgBuf[nImBg]=imgBuf[nImBg+1];
               if(s!=S0){nImCr--;
                     imgBuf[nImBg]=imgBuf[nImCr];nImCr=nImBg+1;}// 04.07.07
               else nImCr=nImBg;}
               return nNode++;}
        else if(ltCls[imgBuf[nImBg]]==dlmaux/*&&ltClsC[imgBuf[nImBg]]>nc*/)
        {imgBuf[nImBg]=imgBuf[nImBg+1];
              nImCr--;}
        return nNode;
   case S1n:// пошук ключових слів та імен
               imgBuf[nImCr-1]=0;
        // пошук у таблиці ключів;
               if(*(imgBuf+nImBg)==13)nImBg++;
              if(pRt){nodes[nNode].ndOp=pRt->func;
        // якщо знайдено
              nodes[nNode].prvNd=NULL;
              nImCr=nImBg;
              if(c!=dlmeormr&&c!=dlmaux)imgBuf[nImCr++]=l;
              return nNode++;}
        // якщо не знайдено
        nodes[nNode].ndOp=_nam;
        insBTr(nodes+nNode,pRtNdx);
        nImBg=nImCr;
        if(c!=dlmeormr&&c!=dlmaux)imgBuf[nImCr++]=l;
        break;
   default:  // не дійшли до класифікованих помилок
   case Eu: case Ec: case Ep: case Eq: case En: case Eo:// обробка помилок
        eNeut(nNode);       // фіксація помилки
   case S1c: case S2c: case S1p: case S2s:     // формування констант
        imgBuf[nImCr-1]=0;
        nodes[nNode].resLength=sP;//      frmCns(sP, nNode); break;
        insBTr(nodes+nNode,pRtNdx);
        if(c!=dlmeormr&&c!=dlmaux)imgBuf[nImCr++]=l;
        break;
   case S3: nImCr=nImBg;
        nodes[nNode].prvNd=NULL;
        imgBuf[nImBg]=imgBuf[nImBg+2];
        s1=s;
        if(s!=S0)nImCr=nImBg+1;//else nImCr--;
  }
  return nNode++;

}
```