Compiler Design(18CSC304J)

Experiment 8

Computation of Follow

Harsh Goel
RA1811003010185

Aim: To study and implement Computation of Follow.

Language: C

Procedure:

1. Create a file or select the file for performing the operations on.
2. Start any python IDE and type the necessary code.
3. Run the code and perform the operations required.
4. Note the output and document it.

Algorithm:

For computing the follow:

1. Always check the right side of the productions for a non-terminal, whose FOLLOW set is being found. (never see the left side).
2. Follow =>
    a. If that non-terminal (S,A,B…) is followed by any terminal (a,b…,*,+,(,)…) , then add that "terminal" into FOLLOW set.
    b. If that non-terminal is followed by any other non-terminal then add "FIRST of other nonterminal" into FOLLOW set.

Code Snippet:

```c
#include<stdio.h>

#include<string.h>

#include<conio.h>

#define max 20
char prod[max][10];
char ter[10], nt[10];
char first[10][10], follow[10][10];
int eps[10];
```

```
int count = 0;
int findpos(char ch) {
  int n;
  for (n = 0; nt[n] != '\0'; n++)
    if (nt[n] == ch)
      break;
  if (nt[n] == '\0')
    return 1;
  return n;
}

int IsCap(char c) {
  if (c >= 'A' && c <= 'Z')
    return 1;
  return 0;
}
void add(char * arr, char c) {
  int i, flag = 0;
  for (i = 0; arr[i] != '\0'; i++) {
    if (arr[i] == c) {
      flag = 1;
      break;
    }
  }
  if (flag != 1)
    arr[strlen(arr)] = c;
}
void addarr(char * s1, char * s2) {
  int i, j, flag = 99;
  for (i = 0; s2[i] != '\0'; i++) {
    flag = 0;
    for (j = 0;; j++) {
      if (s2[i] == s1[j]) {
        flag = 1;
        break;
      }
      if (j == strlen(s1) && flag != 1) {
        s1[strlen(s1)] = s2[i];
        break;
      }
    }
  }
}
void addprod(char * s) {
  int i;
  prod[count][0] = s[0];
  for (i = 3; s[i] != '\0'; i++) {
    if (!IsCap(s[i]))
```

```
      add(ter, s[i]);
    prod[count][i - 2] = s[i];
  }
  prod[count][i - 2] = '\0';
  add(nt, s[0]);
  count++;
}
void findfirst() {
  int i, j, n, k, e, n1;
  for (i = 0; i < count; i++) {
    for (j = 0; j < count; j++) {
      n = findpos(prod[j][0]);
      if (prod[j][1] == (char) 238)
        eps[n] = 1;
      else {
        for (k = 1, e = 1; prod[j][k] != '\0' && e == 1; k++) {
          if (!IsCap(prod[j][k])) {
            e = 0;
            add(first[n], prod[j][k]);
          } else {
            n1 = findpos(prod[j][k]);
            addarr(first[n], first[n1]);
            if (eps[n1] == 0)
              e = 0;
          }
        }
        if (e == 1)
          eps[n] = 1;
      }
    }
  }
}
void findfollow() {
  int i, j, k, n, e, n1;
  n = findpos(prod[0][0]);
  add(follow[n], '$');
  for (i = 0; i < count; i++) {
    for (j = 0; j < count; j++) {
      k = strlen(prod[j]) - 1;
      for (; k > 0; k--) {
        if (IsCap(prod[j][k])) {
          n = findpos(prod[j][k]);
          if (prod[j][k + 1] == '\0') // A -> aB
          {
            n1 = findpos(prod[j][0]);
            addarr(follow[n], follow[n1]);
          }
          if (IsCap(prod[j][k + 1])) // A -> aBb
```

```
          {
            n1 = findpos(prod[j][k + 1]);
            addarr(follow[n], first[n1]);
            if (eps[n1] == 1) {
              n1 = findpos(prod[j][0]);
              addarr(follow[n], follow[n1]);
            }
          } else if (prod[j][k + 1] != '\0')
            add(follow[n], prod[j][k + 1]);
        }
      }
    }
  }
}
void main() {
  char s[max], i;
  printf("\nEnter the productions(type 'end' at the last of the produ
ction)\n");
  scanf("%s", s);
  while (strcmp("end", s)) {
    addprod(s);
    scanf("%s", s);
  }
  findfirst();
  findfollow();
  for (i = 0; i < strlen(nt); i++) {
    printf("%c\t", nt[i]);
    printf("%s", first[i]);
    if (eps[i] == 1)
      printf("%c\t", (char) 238);
    else
      printf("\t");
    printf("%s\n", follow[i]);
  }
  getch();
}
```

Output Screenshots:

```
Enter the productions(type 'end' at the last of the production)
E->TA
A->+TA
A->ε
T->FB
B->*FB
B->ε
F->(E)
F->i
end


NT     First   Follow
_____________________

E        (i        #)
A        +ε        #)
T        (i        +#)
B        *ε        +#)
F        (i        *+#)

_____________________
```

Result:

The code was successfully implemented in C language and output was recorded.