

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
ФАКУЛЬТЕТ ІНФОРМАТИКИ І ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ

РОЗРАХУНКОВО-ГРАФІЧНА РОБОТА
з дисципліни «Системне програмування 2»
Варіант 12

Виконав:
студент 3 курсу
гр. ІО-71
Кривошей Д. А.

Перевірів:
Павлов В. Г.

Київ 2019 р.

Зміст завдання: За основу береться результат виконання лабораторної роботи 6, у якій крім лексичного та синтаксичного аналізу здійснюється семантичний аналіз. У рамках РГР усі перелічені етапи компіляції доповнюються генерацією асемблерного коду, який виконує дії, за допомогою виразу, позначеного у варіанті завдання. Для перевірки працездатності цього коду він повинен бути вбудований у програму на мові високого рівня (мова програмування вибирається самостійно) та уявляти собою лише певну частину обчислення. Усі інші необхідні дії для підготовки та виконання вказаних у завданні обчислень, а також вивід результатів цих обчислень виконуються у програмі на мові високого рівня. При внесенні змін у вираз, який визначається за варіантом завдання, повинен бути згенерований новий асемблерний код.

Таким чином, у межах РГР повинні бути розроблені дві програми:

- одна, яка генерує асемблерний код для заданого виразу, або його варіантів;
- друга, яка використовує згенерований код як частку загальних обчислень, які мають закінчений вигляд та мають бути перевірені не менш ніж п'яти контрольних прикладах з порівнянням з контрольними результатами.

Під час виконання РГР можна використовувати матеріали лабораторних робіт 2.7 та 2.8.

Варіант - 12

Завдання:

12	float b, a[3]; unsigned n,d; b:=2*a[n]; b:=d;	Pascal
----	--	--------

Лістинг програми:

```
1. const addVariablesToTheList = (index) => {
2.   const columnIndex = lexemsTable.indexOf("T_COLUMN", index);
3.   if(columnIndex === -1){
4.     throw new Error(`
5.       \nError at index ${index}, wrong variable declaration \n
6.       Maybe you wanted to place "BEGIN" here?
7.       \n
8.       \n
9.     `)
10.  }
11.  const temporaryVault = [];
12.  let semiColumnIndex = 0;
13.  for(let i = index; i < columnIndex; i++){
14.    if(validVariables.includes(expressionsTable[i])){
15.      console.log({i}, expressionsTable[i]);
16.      temporaryVault.push(expressionsTable[i])
17.    }else if(
```

```

18.     expressionsTable[i] === "," &&
19.     lexemsTable[i + 1] !== "T_VARIABLE"
20.   ){
21.     throw new Error(`
22.       \nError at index ${expressionsIndexes[i]}, you can't use "COMMA" here \n\n
23.     `)
24.   }else if(expressionsTable[i] !== ","){
25.     throw new Error(`
26.       \nError at index ${expressionsIndexes[i]}, you can't use "${expressionsTable[i]}" as a
variable \n\n
27.     `)
28.   }
29.
30. }
31. console.log({columnIndex})
32. if(
33.   lexemsTable[columnIndex+1] !== "T_TYPE" &&
34.   lexemsTable[columnIndex+1] !== "T_ARRAY"
35. ){
36.   throw new Error(`
37.     \nError at index ${expressionsIndexes[columnIndex+1]}, you have to define TYPE here \n\n
38.   `)
39. }
40. if(expressionsTable[columnIndex+1] === "array"){
41.   if(
42.     expressionsTable[columnIndex+2] === "[" &&
43.     lexemsTable[columnIndex+3] === "T_INTEGER" &&
44.     lexemsTable[columnIndex+4] === "T_DOUBLE_DOT" &&
45.     lexemsTable[columnIndex+5] === "T_INTEGER" &&
46.     expressionsTable[columnIndex+6] === "]"
47.   ){
48.     if(
49.       lexemsTable[columnIndex+7] === "T_OF" &&
50.       lexemsTable[columnIndex+8] === "T_TYPE" &&
51.       lexemsTable[columnIndex+9] === "T_SEMICOLUMN"){
52.       semiColumnIndex = columnIndex + 9;
53.       temporaryVault.forEach(e => {
54.         if(declaredVariables[e] === undefined){
55.           declaredVariables[e] = {
56.             "name" : e,
57.             "class": "array",
58.             "type" : `${expressionsTable[columnIndex+8]}`,
59.             "length" : +expressionsTable[columnIndex+5],
60.             "value": Array(+expressionsTable[columnIndex+5]).fill(0)
61.           }
62.         }else{
63.           throw new Error(`
64.             \nError, variable "${e}" has already been declared \n\n
65.           `)
66.         }
67.       })
68.     }
69.     else{
70.       throw new Error(`
71.         \nError at index ${expressionsIndexes[columnIndex+7]}, you have to define TYPE of the
ARRAY correctly \n\n
72.       `)
73.     }
74.   }
75.   else{
76.     throw new Error(`
77.       \nError at index ${expressionsIndexes[columnIndex+2]}, you have to define ARRAY correctly
\n\n
78.     `)
79.   }
80. }else{
81.   if(lexemsTable[columnIndex+2] === "T_SEMICOLUMN"){
82.     semiColumnIndex = columnIndex+2;
83.     temporaryVault.forEach(e => {

```

```

84.     if(declaredVariables[e] === undefined){
85.         declaredVariables[e] = {
86.             "name" : e,
87.             "class":"variable",
88.             "type" :`${expressionsTable[columnIndex+1]}`,
89.             "value":undefined
90.         }
91.     }
92. })
93. }
94. else{
95.     throw new Error(`
96.         \nError at index ${expressionsIndexes[columnIndex+2]}, you have to place ";" after TYPE
97.         \n\n
98.     `)
99. }
100.     console.log(declaredVariables)
101.     console.log({semiColumnIndex})
102.     return semiColumnIndex;
103. }
104.
105. const checkVar = (semiColumnIndex) => {
106.     if(lexemsTable[semiColumnIndex + 1] === "T_BEGIN"){
107.         return false
108.     }else if (lexemsTable[semiColumnIndex + 1] === "T_VARIABLE"){
109.         return true
110.     }else{
111.         throw new Error(`
112.             \nError at index ${expressionsIndexes[semiColumnIndex+1]}, you have to start you
113.             program with "BEGIN" clause\n\n
114.         `)
115.     }
116.
117. const performOperations = (startIndex) => {
118.     let areThereOperations = lexemsTable.includes("T_ASSIGNMENT", startIndex);
119.     let temporaryNumericValue = "";
120.     let temporaryStringValue = "";
121.     let temporaryBooleanValue = "";
122.     let semiColumnIndex ;
123.     let operationIndex = lexemsTable.indexOf("T_ASSIGNMENT", startIndex);
124.     let index = operationIndex+1;
125.
126.     while(areThereOperations){
127.         semiColumnIndex = lexemsTable.indexOf("T_SEMICOLUMN", index);
128.         if(
129.             lexemsTable[operationIndex-1] === "T_VARIABLE" ||
130.             lexemsTable[operationIndex-1] === "T_RIGHT_BRACKET"
131.         ){
132.             let variableToAssign;
133.             if(lexemsTable[operationIndex-1] === "T_RIGHT_BRACKET"){
134.                 console.log("zalupa")
135.                 variableToAssign=declaredVariables[expressionsTable[operationIndex-4]]
136.             }
137.             else{
138.                 variableToAssign=declaredVariables[expressionsTable[operationIndex-1]]
139.             }
140.             if(variableToAssign !== undefined){
141.                 while(index < semiColumnIndex){
142.                     if(
143.                         lexemsTable[index] === "T_VARIABLE"
144.                     ){
145.                         const variable = declaredVariables[expressionsTable[index]];
146.                         if(variable !== undefined){
147.                             if(variable.class === "array"){
148.                                 if(index+2 < variable.length){
149.                                     temporaryNumericValue += variable.value[index]
150.                                     index+=4

```

```

151.         }else{
152.             throw new Error(`
153.                 \nError at index ${expressionsIndexes[index+2]}, out
of "${expressionsTable[index]}" length\n\n
154.             `)
155.         }
156.
157.         }else{
158.             temporaryNumericValue += variable.value;
159.             index++;
160.         }
161.
162.         console.log(temporaryNumericValue);
163.     }else{
164.         throw new Error(`
165.             \nError at index
${expressionsIndexes[index]}, variable "${expressionsTable[index]}" isn't declared\n\n
166.         `)
167.     }
168.     else if(lexemsTable[index] === "T_BINARY_OPERATOR"){
169.         temporaryNumericValue += expressionsTable[index];
170.         index++;
171.         console.log(temporaryNumericValue);
172.     }else if(
173.         lexemsTable[index] === "T_INTEGER" ||
174.         lexemsTable[index] === "T_FLOAT"
175.     ){
176.         temporaryNumericValue += expressionsTable[index];
177.         console.log(temporaryNumericValue);
178.         index++;
179.     }
180. }
181. result = eval(temporaryNumericValue);
182.
183. if(
184.     types[variableToAssign.type].maxValue > result &&
185.     types[variableToAssign.type].minValue < result){
186.     if(variableToAssign.class === "array"){
187.         [operationIndex-2] < variable.length
188.         temporaryNumericValue += variable.value[index]
189.     }else{
190.         variableToAssign.value = result;
191.     }
192.
193. }else{
194.     throw new Error(`
195.         \nError at index ${expressionsIndexes[operationIndex-1]}, variable
"${expressionsTable[operationIndex-1]}" has wrong type\n\n
196.     `)
197. }
198.
199.     console.log(variableToAssign.value);
200. }else{
201.     throw new Error(`
202.         \nError at index ${expressionsIndexes[operationIndex-1]}, variable
"${expressionsTable[operationIndex-1]}" isn't declared\n\n
203.     `)
204. }
205.
206. }
207.
208.     areThereOperations = lexemsTable.includes("T_ASSIGNMENT", index);
209. }
210.
211. }

```

Результати роботи програми:

1.

```
node sp-rgr.js 'var a:array [1..4] of float; b: float; short n, d; a[0] := 10; a[1]:= 20; a[2] := -6; a[3] := 10; n := 10; d := 8;' 'b := 2 * a[0]; b := d;'
result: [ { name: 'b', value: 8 } ]
Assembly code generated.
```

Код асемблера:

```
mov eax, 1092616192
mov dword ptr[a+0], eax // a[0] = 10

mov eax, 1101004800
mov dword ptr[a+4], eax // a[1] = 20

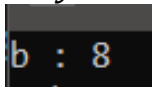
mov eax, 3233808384
mov dword ptr[a+8], eax // a[2] = -6

mov eax, 1092616192
mov dword ptr[a+12], eax // a[3] = 10

mov n, 1092616192 //n = 10
mov d, 1090519040 //d = 8
mov CONSTANT_1, 1073741824 //CONSTANT_1 = 2
mov CONSTANT_2, 1092616192 //CONSTANT_2 = 10

movups xmm0, b
movups xmm1, d
movups xmm2, CONSTANT_1
mov esi, 10
movups xmm3, [4 * esi] + a
mulss xmm2, xmm3
movups b, xmm2
movups b, xmm1
```

Результат асемблерного коду:



b : 8

2.

```
node sp-rgr.js 'var a:array [1..4] of float; b: float; short n, d; a[0] := 10; a[1]:= 20; a[2] := -6; a[3] := 10; n := 10; d := 8;' 'b := 2 * a[0]; d := b;'
```

```
ERROR:
      You try to assign 'b' with 'double' type
      to variable 'd' with 'short' type.
```

3.

```
node sp-rgr.js 'var a:array [1..4] of float; b: float; short n, d; a[0] := 5.5; n := 0; d := 8;' 'b := 2 * a[n];'
result: [ { name: 'b', value: 11 } ]
Assembly code generated.
```

Код асемблера:

```
mov eax, 1085276160
mov dword ptr[a+0], eax // a[0] = 5.5

mov n, 1065353216 //n = 0
```

```

mov d, 1090519040 //d = 8
mov CONSTANT_1, 1073741824 //CONSTANT_1 = 2

movups xmm0, b
movups xmm1, CONSTANT_1
mov esi, 0
movups xmm2, [4 * esi] + a
mulss xmm1, xmm2
movups b, xmm1

```

Результат асемблерного коду:

```
b : 11
```

4.

```

node sp-rgr.js 'var a:array [1..4] of float; b:double; short n, d; a[0] := 5.5; a[1]
:= 5; a[2] := -10; d = 8;' 'b = a[0] + a[1] + a[2];'

```

```

result: [ { name: 'b', value: 0.5 } ]
Assembly code generated.

```

Код асемблера:

```

mov eax, 1085276160
mov dword ptr[a+0], eax // a[0] = 5.5

mov eax, 1084227584
mov dword ptr[a+4], eax // a[1] = 5

mov eax, 3240099840
mov dword ptr[a+8], eax // a[2] = -10

mov d, 1090519040 //d = 8
mov CONSTANT_1, 1065353216 //CONSTANT_1 = 0
mov CONSTANT_2, 1065353216 //CONSTANT_2 = 1
mov CONSTANT_3, 1073741824 //CONSTANT_3 = 2

movups xmm0, b
mov esi, 0
movups xmm1, [4 * esi] + a
mov esi, 1
movups xmm2, [4 * esi] + a
mov esi, 2
movups xmm3, [4 * esi] + a
addss xmm1, xmm2
addss xmm1, xmm3
movups b, xmm1

```

Результат асемблерного коду:

```
b : 0.5
```