Лабораторная работа 16 (4 часа)
Конструирование программного обеспечения

Разработка лексического распознавателя (II часть)

Задание:

1. Используйте материал лекций № 9-13.
2. Используйте результаты лабораторных работ № 13-15.
3. Исследуйте содержимое файла FST.h (рис. 1)

```
namespace FST
{
    struct RELATION       // ребро:символ -> вершина графа переходов КА
    {
     char  symbol;    // символ перехода
     short nnode;     // номер смежной вершины
     RELATION (
               char c = 0x00,    // символ перехода
               short ns = NULL   // новое состояние
               );
    };

    struct  NODE   //  вершина графа переходов
    {
     short n_relation;       // количество инциндентных ребер
     RELATION  *relations;   // инцидентные  ребра
     NODE();
     NODE (
           short n,           // количество инциндентных ребер
           RELATION rel, ...  // список ребер
           );
    };

    struct FST    // недетерминировнный конечный автомат
    {
     char* string;     // цепочка (строка, завершатся 0x00 )
     short position;   // текущая позиция в цепочке
     short nstates;    // количество состояний автомата
     NODE* nodes;      // граф переходов:  [0] -начальное состояние, [nstate-1] - конечное
     short* rstates;   // возможные  состояния автомата на дааной позиции
     FST(
         char* s,      // цепочка (строка, завершатся 0x00 )
         short ns,     // количество состояний автомата
         NODE  n, ...  // список состояний (граф переходов)
         );
    };

    bool execute(         // выполнить распознавание цепочки
                FST& fst  // недетерминировнный конечный автомат
                );
};
```

Рис.1. Содержимое файла FST.h

4. Создайте проект (VS, C++, консольное приложение) с именем **SELab16.**
5. Разработайте конструкторы структур **RELATION**, **NODE,** **FST** и функцию **execute** (табл. 1).

Таблица 1

Наименование функции	**Назначение**
execute	Моделирует работу недетерминированного конечного автомата, разбирающего цепочку символов. Реализует алгоритм разбора цепочки, представленный в лекции 10. **Параметры: fst** - структура (**FST**), описывающая недетерминированный конечный автомат. **Выполняет**: осуществляет разбор цепочки, заданной элементом **string** структуры **FST.** Конечный автомат задан массивом **nodes** структур **NODE**. Первый элемент массива **nodes** описывает вершину **0** графа переходов конечного автомата. Вершина **0** – соответствует начальному (стартовому)состоянию конечного автомата. Последний элемент массива **nodes** соответствует конечному (единственному) состоянию конечного автомата. Количество состояний конечного автомата содержится в элементе **nstates** (тип short) структуры **FST.** Позиция текущего символа в строке **string** хранится в элементе **position** (тип short). Начальное значение **position** равно **0**. В процессе выполнения функции значение **position** увеличивается на **1** в каждой итерации, моделирующей такт работы автомата. В процессе выполнения функции, используется два массива (алгоритм в лекции № 10), размерностями равными **nstates** (количеству состояний автомата). Массив **rstates** в структуре **FST** содержит адрес результирующего массива, после каждой итерации моделирования такта. **Возврат:** тип **bool**. Если разбор цепочки выполнен успешно (автомат разобрал цепочку), то возвращается **true**, иначе **false**. Признаком успешного разбора является значение последнего элемента массива **rstates** равное количеству значимых символов входной цепочки

6. Реализация конструкторов структур и функции **execute**, должна располагаться в пространстве имен **FST** в файле **FST.cpp**
7. Ознакомьтесь с кодом на рис. 2. Здесь приведен пример кода, тестирующего выполнение функции **execute**. При этом применяется автомат, разбирающий цепочки (a+b)*aba. Обратите внимание, каким образом задается конечный автомат (параметры).

```
int _tmain(int argc, _TCHAR* argv[])
{

    setlocale(LC_ALL, "rus");

    FST::FST fst1(              // недетерминированный конечный автомат (a+b)*aba
             "aaabbbaba",                       // цепочка для распознавания
             4,                                 // количество состояний
             FST::NODE(3, FST::RELATION('a', 0), FST::RELATION('b', 0), FST::RELATION('a',1)),  // состояние 0 (начальное )
             FST::NODE(1, FST::RELATION('b', 2)),                                               // состояние 1
             FST::NODE(1, FST::RELATION('a', 3)),                                               // состояние 2
             FST::NODE()                                                                        // состояние 3 (конечное)
                );
    if (FST::execute(fst1))   // выолнить разбор
          std::cout<<"Цепочка "<< fst1.string << " распознана"<< std::endl;
    else  std::cout<<"Цепочка "<< fst1.string << " не распознана"<< std::endl;

    FST::FST fst2(                    // недетерминированный конечный автомат (a+b)*aba
             "aaabbbabba",                      // цепочка для распознавания
             4,                                 // количество состояний
             FST::NODE(3, FST::RELATION('a', 0), FST::RELATION('b', 0), FST::RELATION('a',1)),  // состояние 0 (начальное )
             FST::NODE(1, FST::RELATION('b', 2)),                                               // состояние 1
             FST::NODE(1, FST::RELATION('a', 3)),                                               // состояние 2
             FST::NODE()                                                                        // состояние 3 (конечное)
                );
    if (FST::execute(fst2))   // выолнить разбор
          std::cout<<"Цепочка "<< fst2.string << " распознана"<< std::endl;
    else  std::cout<<"Цепочка "<< fst2.string << " не распознана"<< std::endl;

    system("pause");
    return 0;
}
```

Рис.2. Пример кода для тестирования функции **execute**.

8. По аналогии рис. 2 разработайте код для проверки семи цепочек символов, подготовленных в рамках лабораторной работы 13.
9. Предложите цепочку, при которой разбор проходит все символы входной цепочки, но при этом, цепочка не распознается.
10. Предложите цепочку, при которой разбор завершается, не перебрав все символы входной цепочки.
11. Добавьте в тестирующий код для разбора 2-х цепочек, не распознающихся разработанным конечным автоматом.
12. В результате выполнения лабораторной работы должно быть разработано приложение, состоящее из файлов FST.h (рис. 1), FST.cpp (конструкторы структур и функция execute), а также файл SELab16.cpp, содержащий 9 (7+2) примеров, тестирующих программу.
13. Продемонстрируйте работу алгоритма разбора цепочки символов, основанного на двух массивах, на конкретном примере.

Вопросы:

1. Как называется первая фаза компилятора.
2. Дайте определение лексического анализа.
3. Какие функции выполняет лексический анализатор.
4. Какая грамматика по иерархии Хомского применяется для описания лексики языка программирования.
5. Дайте определение регулярного выражения.
6. Дайте определение конечного автомата.
7. Приведите схему работу лексического анализатора.
8. Соотношение регулярного языка, регулярной грамматики, регулярного языка и конечного автомата.
9. Конечный автомат: определение графа переходов конечного автомата и метод его построение по регулярному выражению.
10. Конечный автомат: алгоритм разбора цепочки символов, основанный на двух массивах.