```
expr            /* <выр> */
{
  t;
  while (nx="+") do
      begin
          scan;
          t;
      end;
}
/*-------------------------------------------------------------------------*/
t               /*     T     */
{
  o;
  while (nx="*") do
      begin
         scan;
         o;
      end;
}
/*-------------------------------------------------------------------------*/
o;              /*     O     */
{
  if (nx="(") then
      begin
         scan;
         expr;
         if (nx!=")") then error;
         else scan;
      end;
  else var;
}
```

5.5. ДИАГНОСТИКА И НЕЙТРАЛИЗАЦИЯ СИНТАКСИЧЕСКИХ ОШИБОК

Диагностика – установка места возникновения и типа синтаксической ошибки. Кроме того, обработанная ошибка должна быть визуализирована пользователю в виде, удобном для ее обнаружения.

Нейтрализация – предполагает исключение синтаксически неверной конструкции в тексте безболезненно для дальнейшего разбора всего текста.

Систематических методов нейтрализации не существует, поэтому в каждом языковом процессоре разработчику предоставляется самостоятельная задача по нейтрализации ошибки.

Айронс в 1968 г. предложил метод [1] локализации и отсечения «больных» кустов дерева при нисходящем разборе программы. Метод не претендует на универсальность, однако в нем выработаны здравые концепции нейтрализации ошибок при нисходящем разборе.

МЕТОД АЙРОНСА

Основная идея – по контексту без возврата отбрасывать литеры, которые привели к тупиковой ситуации (когда продолжение анализа по грамматике невозможно), и продолжать разбор. Для иллюстрации метода рассмотрим грамматику **G**[*P*]:

$$P \to A$$
$$A \to i = E$$
$$E \to T\{+T\}$$
$$T \to O\{*O\}$$
$$O \to i \mid (E).$$

Представленная грамматика **G**[*P*] является усечением арифметического оператора присваивания, причем усечение выполнено на уровне операнда и операций. Это не нарушает общности грамматики и сделано для того, чтобы не увеличивать размерность синтаксического дерева.

Пример

Пусть оператор присваивания задан строкой $i = i +)$. Этот оператор присваивания явно не соответствует грамматике **G**[*P*]. Построим синтаксическое дерево для этой основы (рис. 5.20).

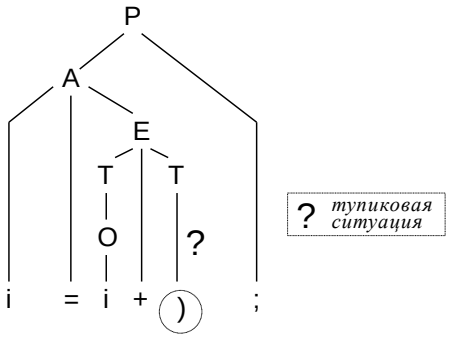

Рис. 5.20. Дерево с диагностикой ошибки

В общем случае обнаружение ошибки соответствует следующей схеме разбора:

$$Z =>^* x_1x_2 \ldots x_{i-1}x_i \ldots x_n,$$

где $x_1x_2 \ldots x_{i-1}$ – построенная часть куста; $x_i \ldots x_n$ – недостроенная часть куста, которую нельзя построить в $\mathbf{G}[Z]$.

Часто при диагностике в тупиковой ситуации на экран выводится недостроенная часть или «хвост» сентенциальной формы. Задачей проектировщика процессора ошибок является нейтрализация ошибки с целью дальнейшего разбора и возможного исправления допущенной ошибки.

АЛГОРИТМ АЙРОНСА ПО ИСПРАВЛЕНИЮ ОШИБОК

Пусть xjy – куст исходной программы, где x – построенная часть, jy – недостроенная часть, $j \in \mathbf{V_T}$.

1. Строим список L из литер недостающих частей неполных кустов.

2. Головной терминальный символ j в цепочке jy проверяется и отбрасывается (при этом каждый раз получается новая цепочка jy) до тех пор, пока не найдется такой символ j, что будет иметь место $U =>^* j...$ (выводимость).

3. Определяется неполный куст, ставший причиной появления ошибки.

4. Определяется терминальная цепочка q. Если ее поставить перед j, то продолжение разбора приведет к правильной привязке к неполному кусту, найденному на шаге 3, и всем кустам поддерева. Для каждого такого куста генерируется цепочка терминалов до полного поддерева, а конкатенация этих цепочек дает q.

5. Цепочка q вставляется непосредственно перед j, и разбор продолжается, начиная с головного символа цепочки q, который становится входным.

Для нашего примера цепочка $X \equiv \{i = i +\}$

$j \equiv)$ $jy \equiv);$

$xjy, j \in V_T$

1) L = {;, T, +}

2) $j \equiv);$ $P \to A.$

Необходимо вставить цепочку q, чтобы дополнить $E \to T\{+T\}$, при этом проще всего в качестве цепочки q рассмотреть символ i ($q = i$).

Тогда дерево будет иметь следующий вид (рис. 5.21).

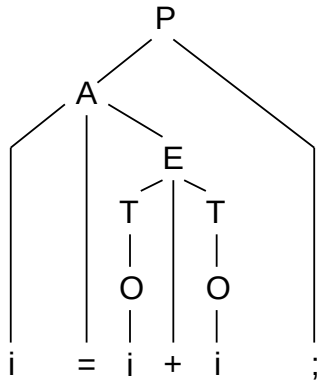

Рис. 5.21. Дерево с нейтрализацией ошибки

5.6. АНАЛИЗ АРИФМЕТИЧЕСКИХ ВЫРАЖЕНИЙ

Вычисление арифметических выражений <AB> происходит в два этапа. На первом этапе обычная традиционная, или *инфиксная*, форма записи <AB> преобразуется в *постфиксную*, или *польскую инверсную* запись (ПОЛИЗ), названную так в честь польского математика Яна Лукасевича. На втором этапе ПОЛИЗ преобразуется в результат <AB>. Примеры инфиксной и постфиксной записи арифметических выражений приведены в табл. 5.1.

Таблица 5.1

Инфиксная	Постфиксная
*a*b + c*	*ab*c +*
*a + b * c*	*abc* +*
*a + b * c – d/(a + b)*	*abc* + dab + /–*

Отличие ПОЛИЗ от инфиксной формы состоит в отсутствии круглых скобок и порядке следования операндов и операций. Вычислять выражения на основе ПОЛИЗ значительно проще, так как отсутствие скобок, переопределяющих приоритет выполнения операций, исключает дополнительный алгоритм порядка выполнения