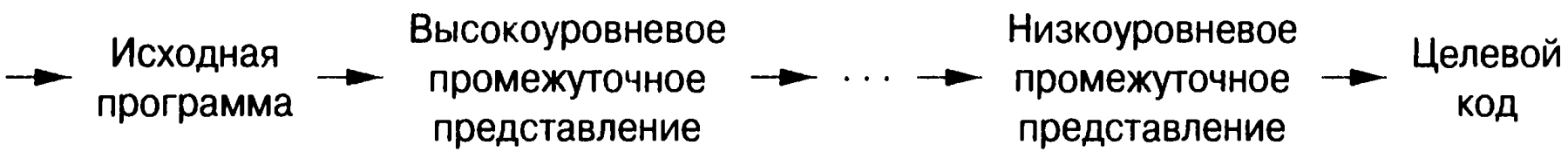
Способы внутреннего представления программ

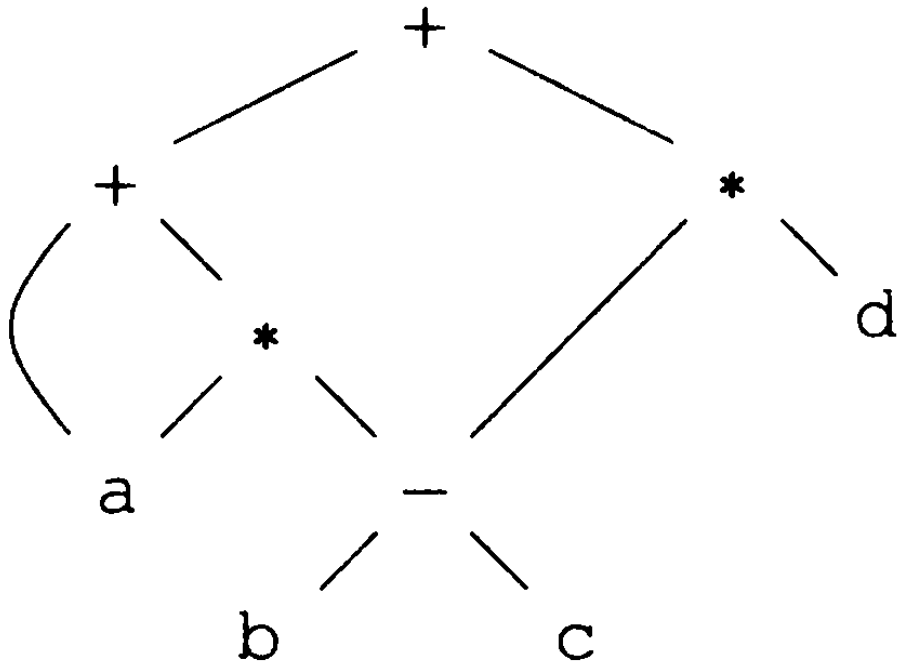
- ✓ Связные списочные структуры, представляющие синтаксические деревья
- ✓ Многоадресный код с явно именуемым результатом (тетрады)
- ✓ Многоадресный код с неявно именуемым результатом (триады)
- ✓ Постфиксная (польская инверсная) запись



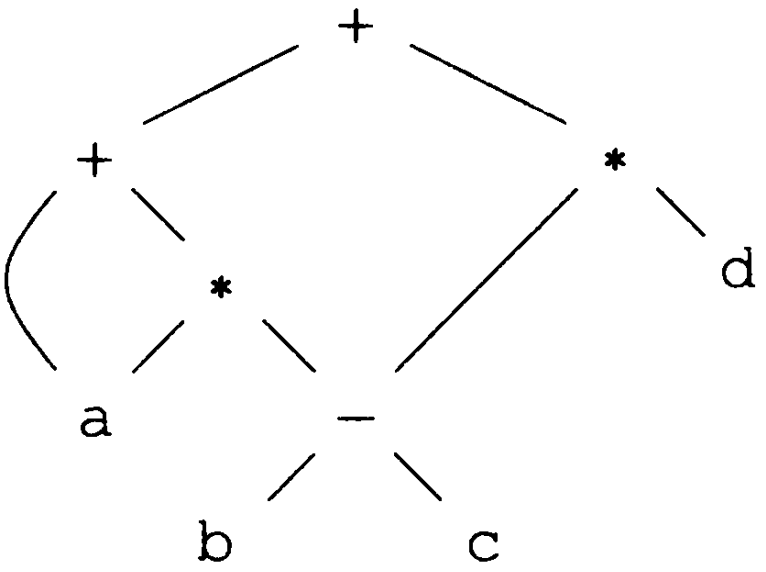
Связные списочные структуры

- Ориентированный ациклический граф имеет листья, соответствующие атомарным операндам, и внутренние узлы, соответствующие операторам

```
a+a*(b-c)+(b-c)*d
```



Ориентированный ациклический граф и трехадресный код



$t_1 = b - c$
$t_2 = a * t_1$
$t_3 = a + t_2$
$t_4 = t_1 * d$
$t_5 = t_3 + t_4$

Трехадресные команды

- Команды присваивания

  `x = y op z`

  `op` — бинарная арифметическая или логическая операция,

  `x`, **`y`** и **`z`** — адреса

  `x = op y`

  `op` — унарная операция

- Команды копирования

  `x = y`

- Безусловный переход

  `goto L`

- Условный переход

  `if x goto L`

  `ifFalse x goto L`

  `if x relop y goto L`

Трехадресные команды

- Вызовы процедур и возврат из них

`param x` — передача параметров

`call p, n`

`y = call p, n`

} вызов процедуры или функции

`return y`

`y` — необязательное возвращаемое значение

$$\begin{aligned}
&\texttt{param}\ x_1\\
&\texttt{param}\ x_2\\
&\ldots\\
&\texttt{param}\ x_n\\
&\texttt{call}\ p, n
\end{aligned}$$

Трехадресные команды

- Индексированные присваивания

```
x = y[i]
x[i] = y
```

- Присваивание адресов и указателей

```
x = &y
x = *y
*x = y
```

Пример

```
do i = i+1; while (a[i] < v);
```

```
L:   t1 = i + 1
     i = t1
     t2 = i * 8
     t3 = a [ t2 ]
     if t3 < v goto L
```

```
100:  t1 = i + 1
101:  i = t1
102:  t2 = i * 8
103:  t3 = a [ t2 ]
104:  if t3 < v goto 100
```

Тетрады

- *Тетрада* имеет четыре поля: **op**, **arg1**, **arg2** и **result**
- Поле **op** содержит внутренний код оператора.

  Например, для x = y + z

  op = +, **arg1** = y, **arg2** = z и **result** = x
- Команды с унарными операторами наподобие x = minus y или x = y не используют **arg2**. В случае команды копирования x = y поле op равно =, в то время как для большинства других операций оператор присваивания подразумевается неявно
- Условные и безусловные переходы помещают в поле **result** целевую метку

Пример

```
a=b*-c+b*-c;
```

t_1 = minus c
t_2 = b * t_1
t_3 = minus c
t_4 = b * t_3
t_5 = t_2 + t_4
a = t_5

	op	arg_1	arg_2	*result*
0	minus	c		t_1
1	*	b	t_1	t_2
2	minus	c		t_3
3	*	b	t_3	t_4
4	+	t_2	t_4	t_5
5	=	t_5		a
	...			

Триады

- *Триады* состоят только из трех полей — op, arg1 и arg2.

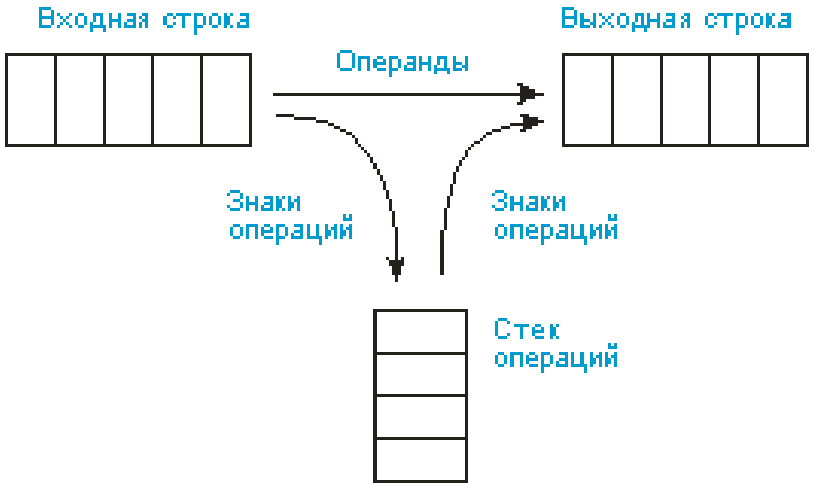
```
a=b*-c+b*-c;
```

	op	arg_1	arg_2
0	minus	c	
1	*	b	(0)
2	minus	c	
3	*	b	(2)
4	+	(1)	(3)
5	=	a	(4)
	...		

Польская инверсная запись (ПОЛИЗ)

Ян Лукашевич
(1878 – 1956)

Обратная польская нотация была разработана австралийским философом и специалистом в области теории вычислительных машин Чарльзом Хэмблином в середине 1950-х на основе польской нотации, которая была предложена в 1920 году польским математиком Яном Лукашевичем



Чарльз Леонард Хэмблин
(1922 – 1985)

Алгоритм записи выражения в ПОЛИЗ

Просматриваем инфиксную запись слева направо

Если встречаем операнд, то копируем в постфиксную запись

Если встречаем знак операции, то

Если стек пуст, то помещаем знак операции в стек

Иначе

Если приоритет последней операции в стеке < приоритета текущей операции, то помещаем знак операции в стек

Иначе

Записываем знаки операций из стека в постфиксную запись до тех пор, пока приоритет последней операции в стеке >= приоритету текущей операции

Помещаем знак текущей операции в стек

Записываем знаки операций из стека в постфиксную запись до тех пор, пока стек не пуст

Пример

6 + 7 + 10 * 4

Выражение в ПОЛИЗ	Стек операций
6	+
67+	+
67+10	* +
67+104*+	

Алгоритм вычисления выражения в ПОЛИЗ

Просматриваем польскую запись слева направо

Если встречаем операнд, то помещаем его в стек

Если встречаем знак операции,

то выполняем эту операцию, используя в качестве операндов два числа с вершины стека.

Результат помещаем в стек.

Пример

67+104*+

Входные символы	Стек операндов
6	
7	6
+	7 6
10	13
4	10 13
*	4 10 13
+	40 13
	53

Алгоритм Дейкстры

Эдсгер Дейкстра (1930 - 2002) разработал алгоритм для преобразования выражений из инфиксной нотации в ПОЛИЗ

Алгоритм получил название *алгоритм сортировочной станции* из-за сходства его операций с происходящим на железнодорожных сортировочных станциях

Пока не все токены обработаны:

Прочитать токен.

Если токен — *число*, то добавить его в очередь вывода.

Если токен — *функция*, то поместить его в стек.

Если токен — *разделитель аргументов функции* (например запятая):

Пока токен на вершине стека не *открывающая скобка*, перекладывать операторы из стека в выходную очередь. Если в стеке не было *открывающей скобки*, то в выражении пропущен *разделитель аргументов функции* (запятая), либо пропущена *открывающая скобка*.

Если токен — *оператор* op1, то:

Пока присутствует на вершине стека токен *оператор* op2, и

Либо *оператор* op1 левоассоциативен и его приоритет меньше чем у *оператора* op2 либо равен,

или *оператор* op1 право ассоциативен и его приоритет меньше чем у op2,переложить op2 из стека в выходную очередь;

положить op1 в стек.

Если токен — *открывающая скобка*, то положить его в стек.

Если токен — *закрывающая скобка*:

Пока токен на вершине стека не является *открывающей скобкой*, перекладывать операторы из стека в выходную очередь.

Выкинуть *открывающую скобку* из стека, но не добавлять в очередь вывода.

Если токен на вершине стека — функция, добавить ее в выходную очередь.

Если стек закончился до того, как был встречен токен *открывающая скобка*, то в выражении пропущена скобка.

Если больше не осталось токенов на входе:

Пока есть токены операторы в стеке:

Если токен оператор на вершине стека — *открывающая скобка*, то в выражении присутствует незакрытая скобка.

Переложить оператор из стека в выходную очередь.

Конец.

Представление операторов в ПОЛИЗ

I := E

В ПОЛИЗ будет записан как

I E :=

где ":=" – это двухместная операция,
а I и E – ее операнды;

I означает, что операндом операции ":="
является адрес переменной I, а не ее значение.

Расширение набора операций ПОЛИЗ

- *Операция перехода (обозначается «!»)* в терминах ПОЛИЗ означает, что процесс интерпретации надо продолжить с того элемента ПОЛИЗ, который указан как операнд этой операции.

- Чтобы можно было ссылаться на элементы ПОЛИЗ, будем считать, что все они перенумерованы, начиная с 1 (например, занесены в последовательные элементы одномерного массива).

- Пусть ПОЛИЗ оператора, помеченного меткой L, начинается с номера p, тогда оператор перехода

goto L

в ПОЛИЗ записывается так:

p !

где ! – операция выбора элемента ПОЛИЗ, номер которого равен p.

Расширение набора операций ПОЛИЗ

- *Операция условного перехода* с семантикой

  if (!B) goto L

  это двухместная операция с операндами B и L.
- Обозначим ее

  !F

  тогда в ПОЛИЗ переход записывается так:

  B’ p !F

  где p — номер элемента, с которого начинается ПОЛИЗ оператора, помеченного меткой L, B’ — ПОЛИЗ логического выражения B.
- Семантика *условного оператора*

  if E then S1 else S2

  с использованием введенной операции может быть описана так:

  if (!E) goto L2; S1; goto L3; L2: S2; L3: ...

  Тогда ПОЛИЗ условного оператора будет таким (порядок операндов — прежний):

  E’ p2 !F S1’ p3 ! S2’ ... ,

  где pi – номер элемента, с которого начинается ПОЛИЗ оператора, помеченного меткой Li, i = 2,3, E’ – ПОЛИЗ логического выражения E.

Расширение набора операций ПОЛИЗ

- Семантика оператора *цикла*

while E do S

может быть описана так:

L0: if (!E) goto L1; S; goto L0; L1: ... .

Тогда ПОЛИЗ оператора цикла while будет таким:

E’ p1 !F S’ p0 ! ... ,

где pi - номер элемента, с которого начинается ПОЛИЗ оператора, помеченного меткой Li,

i = 0,1,

E’ – ПОЛИЗ логического выражения E.

Пример

- Записать в ПОЛИЗе следующий оператор языка Си, используя стек, выполнить его при указанных начальных значениях переменных:

```
S = 0;
for (I = 1; I <= k; I = I + 1)
S = S + i*i;
```

при k = 3.

Пример (окончание)

```
S = 0;
I = 1;
k = 3;
while(I <= k)
{
  S = S + i*i;
   I = I + 1;
}
```

0	1	2	3	4	5	6	7	8
S0=	I1=	k3=	Ik<=	p8!F	SSii*+=	II1+=	p3!	…

Пример

- Записать в ПОЛИЗе следующий оператор языка Си, используя стек, выполнить его при указанных начальных значениях переменных:

```
switch (k)
{
  case 1: a = not a; break;
  case 2: b = a or not b; break;
  case 3: a = b; break;
}
```

Пример (окончание)

```
if(k == 1)
{
  a = not a;
}else if(k == 2)
{
   b = a or not b;
}else if(k == 3)
{
   a = b;
}
```

0	1	2	3	4	5	6	7	8	9	10
k1==	p4!F	aanot=	p11!	k2==	p8!F	babnotor=	p11!	k3==	p11!F	ab=

Расширение набора операций ПОЛИЗ

- *Операторы ввода и вывода* являются одноместными операциями.
- *Оператор ввода*

read (I)

в ПОЛИЗе будет записан как

I read

- *Оператор вывода*

write (E)

в ПОЛИЗе будет записан как

E’ write

где E’ – ПОЛИЗ выражения E.