§ 6. Проверка семантической корректности программы – общие сведения. Выражения типа

Компилятор проверяет синтаксическую правильность программы. Кроме того, он выполняет часть проверки семантической корректности. Более точно, компилятор выявляет те ошибки, которые можно обнаружить без запуска программы.

Такую проверку называют статической.

Список проверок:

1. Проверка типов.
2. Проверка управления.
3. Проверка единственности.
4. Проверка имён.

Динамической проверкой называется проверка в ходе выполнения программы. Например, деление на 0, переполнение памяти, обращение к неопределенным динамическим объектам, обращение к массиву с неверным индексом.

В нашем курсе обсуждается только случай статической проверки.

2. Проверка управления.

Пример. Переход от одной точки в программе в другую, расположенную внутри цикла – ошибка.

Проверка управления подтверждает, что передача управления в программе осуществляется в существующее и разрешённое место.

3. Проверка единственности.

В большинстве языков программирования разрешается лишь однократное объявление имён (идентификаторов, функций).

4. Проверка имен.

Пример. Название функции, совпадающее с ключевым словом, соответствующим команде – может привести к ошибке.

1. Проверка типов.

Любой элемент данных в программе имеет свой тип. От типа зависит структура элемента, способ представления в памяти, объем памяти для хранения, допустимые методы обработки. Также от типа данных зависит семантика, т.е. смысл данных.

Тип данных задается «выражением типа».

Выражение типа получается из элементарных (базовых) типов применением «конструкторов типа», т.е. операторов, операндами которых могут быть выражения типа (рекурсивное описание выражения типа).

Любое выражение типа имеет «имя типа».

Назовем переменными типа – переменные, хранящие выражение типа.

! Базовые типы не имеет внутренней структуры.

Далее базовыми типами будут:

1) int
2) real
3) char
4) bool
5) type-error
6) void

} константы типа

Далее конструкторами типов будут:

1) Массив.
2) Произведение.
3) Запись.
4) Указатель.
5) Функция.

(Рекурсивное определение выражения типа.)
Опр. 1. Константа типа, имя типа, переменная типа являются выражением типа.

2.1. Если T – выражение типа, N – натуральное число, то array(N, T) – выражение типа. Результат применения конструктора – массив длины N элементов типа T. Индексация – от 0 до (N–1).

2.2. Если T_1, T_2 – выражения типа, то $T_1 \times T_2$ является выражением типа. Результат применения конструктора – упорядоченная пара элементов типа T_1 и T_2 (элемент декартова произведения).

2.3. Если T_1, T_2, …, T_k – выражения типа, $name_1$, $name_2$, …, $name_k$ – их имена, то record(($name_1$×T1)×($name_2$×T_2)×($name_k$×T_k)) – выражение типа. Результат применения конструктора – тип записи с полями T_1, T_2, …, T_k, имеющими имена $name_1$, $name_2$, …, $name_k$. Имя типа записи задается при объявлении.

2.4. Если T – выражение типа, то ptr(T) является выражением типа. Результат применения конструктора – тип «указатель на объект типа T».

2.5. Если T_1, T_2 – выражения типа, то $T_1 \rightarrow T_2$ является выражением типа. Результат применения конструктора – функция (отображение) из множества элементов типа T_1 в множество элементов типа T_2.

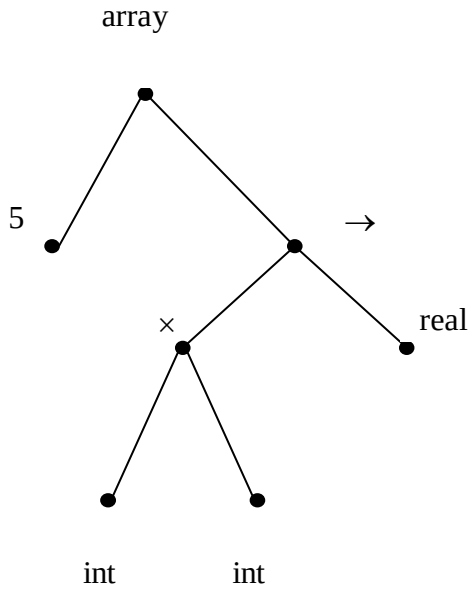
3. Других выражений типа нет.

Комментарий: рекурсивное определение выражения типа – КС-грамматика. И эта грамматика – операторная.

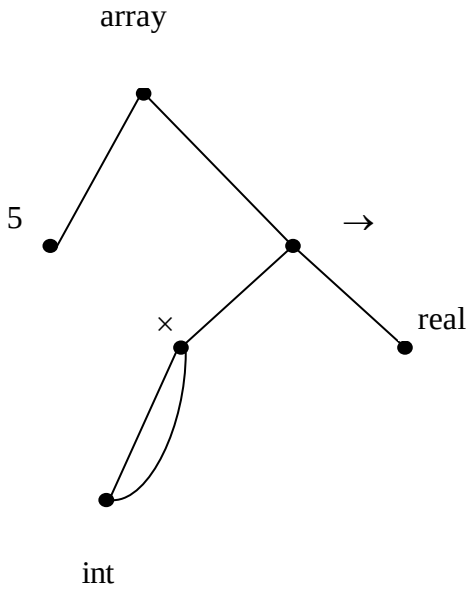
Выражение типа можно представить в виде синтаксического дерева, либо в виде дага.

Пример. array(5, (int×int)→real))

Пример. array(5, (int×int)→real))



Синтаксическое дерево



Даг

Опр. Выражения типа называются структурно эквивалентными, если их даги изоморфны.