0.1 Tipi

I tipi implementati nel linguaggio sono:

- `bool`: rappresenta un tipo booleano `true` o `false`;
- `int`: rappresenta un tipo intero;
- `pointer`: rappresenta un tipo puntatore. Questo può puntare ad un altro puntatore oppure ad un tipo primitivo quindi booleano o intero;
- `void`: rappresenta il tipo vuoto;
- `arrowType`: tipo utilizzato per le *funzioni*, contiene la lista dei tipi dei parametri e il tipo ritornato.

Particolare attenzione viene rivolta verso il corretto uso dei puntatori e la conformità dei parametri formali con gli attuali.

0.1.1 Corretto uso dei puntatori

Il corretto uso dei puntatori è definito da alcuni punti cruciali:

- Tutte le referenze di un puntatore devono essere inizialiazzate per poter essere scritte/lette;
- Prima di usare un puntatore cancellato questo va reinizializzato;
- Non è possibile accedere ad un puntatore cancellato;
- Il livello di dereferenziazione durante l’assegnamento deve combaciare.

Riguardo al controllo sull’inizializzazione rimandiamo alla sezione dell’analisi degli effetti.

0.1.2 Parametri attuali non conformi ai parametri formali

Il controllo sui parametri di una funzione avviene in due parti, la prima controlla che il numero dei parametri sia lo stesso di quello presente nella dichiarazione della funzione invocata ed inseguito viene controllato che anche i tipi combacino.

```
public TypeNode typeCheck () throws SimplanPlusException {
    List<TypeNode> p = t.getParList();
    if ( !(p.size() == parameterlist.size()) )
        throw new SimplanPlusException('Wrong number of parameters in the
            invocation of '+id);

    for (int i=0; i<parameterlist.size(); i++)
    if ( !(TypeUtils.isSubtype( (parameterlist.get(i)).typeCheck(), p.get(i)) )
        )
        throw new SimplanPlusException('Wrong type for '+(i+1)+'-th parameter
            in the invocation of '+id);
}
```