* parametri attuali non conformi ai parametri formali (inclusa la verifica sui parametri passati per var) * la correttezza dei tipi

0.1 Variabili o Funzioni non dichiarate

```
{
    int x = c;
}
```

In questo caso viene sollevato un errore semantico in quando l’identificatore c non è presente all’interno della symbol table.

```
{
    int x = f();
}
```

Questo invece è un caso analogo al primo in cui viene sollevato lo stesso errore dovuto al fatto che la funzione non è dichiarata.

In entrambi i casi l’output è: *Id ’id’ not declared.*

0.2 Dichiarazioni multiple

```
{
    int x = 1;
    bool x = true;
}
```

Qui viene sollevato un errore in quando la variabile con identificatico x è già presente nello scope attuale.

```
{
    void x(){
        print 1;
    }
    void x(){
        print 2;
    }
}
```

Come nel caso precedente viene sollevato un errore semantico: *Var—Fun id ’x’ already declared*

0.3 Variabili non inizializzate

```
{
    int x;
    int y;
    x = y;
}
```

In questo caso viene sollevato un errore a run time in quanto in base alle slide della lezione un identificatore presente nel rhs viene messo in automatico ad uno stato rw. L’analisi degli effetti non produce quindi nessun errore ma a run time il programma si blocca dicendo che il valore non è scritto in memoria.

```
{
    ^int x;
    int y;
    x^ = y;
}
```

Questo programma genera un errore durante la compilazione perchè il puntatore x non ha tutte le dereferenziazioni precedenti a rw.

```
{
    ^int x = new int;
    int y;
    x^ = y;
}
```

In questo caso viene restituito un errore a run time come nel primo caso. *Value is not written in memory*

0.4 Puntatori

```
{
    ^int x = new int;
    ^^int y = new ^int;
    y^ = new int;
    y^^ = 2;
    x^ = y^^ + 1;
    print x^;
    x = y^;
    print x^;
}
```

Questo programma compila ed esegue correttamente facendo notare il corretto utilizzo dei puntatori. Le dereferenziazioni corrispondono, i puntatori sono inizializzati correttamente e quindi non viene sollevato alcun errore statico. Inoltre i valori sono effettivamente scritti in memoria e quindi non viene sollevato neanche un errore a run time. Il programma termina correttamente stampando prima 3 e poi 2, in quando x viene riassegnato alla stessa zona di memoria di y.

0.5 Conformità dei parametri

```
{
    void sum(int a, int b){
        print a+b;
    }
    sum(1);
}
```

Questo programma genera un errore semantico in quanto il numero dei parametri attuali non è conforme ai parametri formali. Analogalmente il seguente codice:

```
{
    void sum(int a, int b){
        print a+b;
```

```
    }
    sum(1,true);
}
```

Genera un altro errore semantico: *Wrong type for 2-th parameter in the invocation of sum* in quanto il secondo parametro attuale non è dello stesso tipo del secondo parametro formale.

```
{
    void sum(int a, int b){
        print a+b;
    }

    ^int x = new int;
    x^ = 1;
    sum(x,2);
}
```

Infine in questo caso viene sollevato lo stesso errore semantico del secondo caso in quanto la variabile non viene dereferenziata in conformità con il parametro formale.

0.6 Correttezza dei tipi

```
{
    ^int x = new int;
    int c = 10;

    bool boolean = true;
    ^bool pointerBoolean = new bool;

    print c;
    print boolean;

    x^ = 2;
    pointerBoolean^ = false;
    c = x^;
    boolean = pointerBoolean^;

    print c;
    print boolean;

}
```

Il seguente programma dimostra la correttezza dei tipi implementata. Per entrambe le variabili la dereferenziazione avviene correttamente e i tipi combaciano. Il programma stampa prima i valori assegnati durante la dichiarazione ovvero 10 e true(1). Dopo l’assegnamento l’output è 2 e false(0).