Type Checking

Dato un programma il type checker deve verificare che i tipi delle espressioni siano quelli attesi. Se il tipo delle espressione non è quello atteso scrive in output un warning o un messaggio di errore (viene specificata sotto la differenza).
Possiamo assumere che nei programmi da tipare ci siano solo tre tipi: *void*, *int* e *float*.
Inoltre si facciano le seguenti ipotesi:

I tipi delle espressioni sono originati

- da costanti (5 è di tipo *int* e 6.31 è di tipo *float*),
- da parametri formali (x in void f (int x) {...}) e
- dalle dichiarazioni di variabili all'inizio dei blocchi ({int x, float y; ...}).

I tipi delle espressioni semplici vengono poi propagati a espressioni più complesse (se il tipo di x ed y è *int* il tipo di x + y è ancora *int*).
Le dichiarazioni delle variabili sono solo all'inizio dei blocchi, quindi non ci sono dichiarazioni di variabili globali e non ci sono dichiarazioni in blocchi annidati.
Si può anche assumere che un programma sia costituito da una definizione unica di funzione.

Il tipo previsto di un'espressione può essere determinato da

(1) operatori e sottoespressioni (per esempio in x + y, x e y dovrebbero essere dello stesso tipo),
(2) il tipo del lato sinistro in un assegnamento (in x = y, y dovrebbe avere lo stesso tipo di x),
(3) la dichiarazione di controllo in cui si verifica un'espressione (le guardie booleane in istruzioni di controllo dovrebbero avere tipo int) e
(4) tipi nel return delle funzioni (in int f (...) {... return x;}, x dovrebbe avere tipo int).

Il Type Checker deve essere in grado di gestire

- sequenze di istruzioni,
- if,
- while,
- assegnamenti,
- return,
- espressioni che contengono:
    - costanti,
    - variabili,
    - (,), ~, -, +, *, /, < ,>, + +, -, ==, =, <=,> =, &&, | | e
    - il cast di tipo (int) e (float).

Per gli operatori booleani ci aspettiamo che gli operandi siano di tipo int; per gli operatori numerici che gli operandi siano dello stesso tipo (tranne ++ e -- che funzionano solo con tipo int).
Dobbiamo gestire anche i cast ad altri tipi: *(int)* e *(float)* (modificano il tipo di un'espressione da *float* ad *int* e viceversa.)
Il tipo di un *return* può essere nullo, il che significa che possiamo avere funzioni che ritornano *void*.

Un errore di tipo si verifica quando il tipo di un'espressione non è lo stesso del tipo previsto. Si verifica un WARNING quando si prevede un *float*, ma è dato un *int*. Tutti gli altri casi generano un ERROR.

Esempio 1.

```
 float f(int a){
int x;
 float y;
x=12;
 y=23;
x=3.78;
return 0;
}
```

warning: type cast int to float, line 5
type error: int expression expected, line 6

Esempio 2.

```
 float f(int a){
int x;
 x=x + a * (int)3.4;
 while(3.5){
x=3.78;
 };
 }
```

type error: int expression expected, line 4

Esempio 3.

```
 void main(int a){
int x;
 float z;
 void y;
 z= x + 1 + x;
 if(y==y){
x=3.78;
 };
 }
```

warning: type cast int to float, line 5
type error: numerical expression expected, line 6

Esempio 4.

```
void main(int a){
int x;
 void y;
 x=a * 3.4;
 if(y==y){
x=3.78;
 };
 }
```

warning: type cast int to float, line 4
type error: int expression expected, line 4

Esempio 5.

```
int f(){
return 1.1;
}
```

type error: int expression expected, line 2