Da infissa a Postfissa

Facciamo vedere come si può convertire un'espressione aritmetica infissa in una equivalente postfissa, come secondo esempio di uso della pila e per completare il nostro primo caso di uso, quello in cui valutavamo un'espressione aritmetica postfissa. Così combinando i due esempi otteniamo un programma di valutazione di espressioni aritmetiche infisse.

Per trasformare un'espressione infissa in una postfissa notiamo innanzi tutto che gli operandi devono comparire nella notazione postfissa nello stesso ordine che in quella infissa. Quindi se leggiamo la stringa da sinistra verso destra ogni operando viene semplicemente accodato nella stringa postfissa.

Per gli operatori invece bisogna tener conto delle regole di precedenza. Infatti a+b*c viene tradotto in abc*+, nella quale il prodotto è calcolato prima della somma, ma (a+b)*c deve essere tradotto in ab+c*, in modo che si calcoli prima la somma e poi il prodotto.

Regole di precedenza

(non ha precedenza su alcun operatore

ma anche nessun operatore ha precedenza su (

Invece ogni operatore, tranne (, ha precedenza su)

* e / hanno precedenza su + e -

algoritmo per trasformare un'espressione infissa in un postfissa

Leggi la stringa da sinistra verso destra e

1. se il simbolo letto è un operando, accodalo alla postfissa

2. altrimenti, se la pila non è vuota [il simbolo letto è un operatore], e se la precedenza dell'operatore in cima alla pila è maggiore di quello letto, estrai dalla pila l'operatore e accodalo all'espressione postfissa

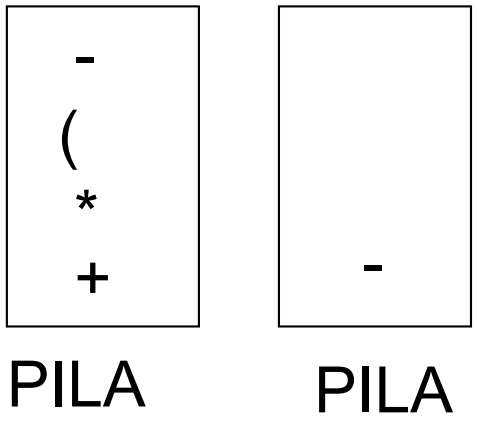
se il simbolo in lettura è una ')' estrai dalla pila senza accodare [in cima alla pila c'e' la parentesi aperta corrispondente]

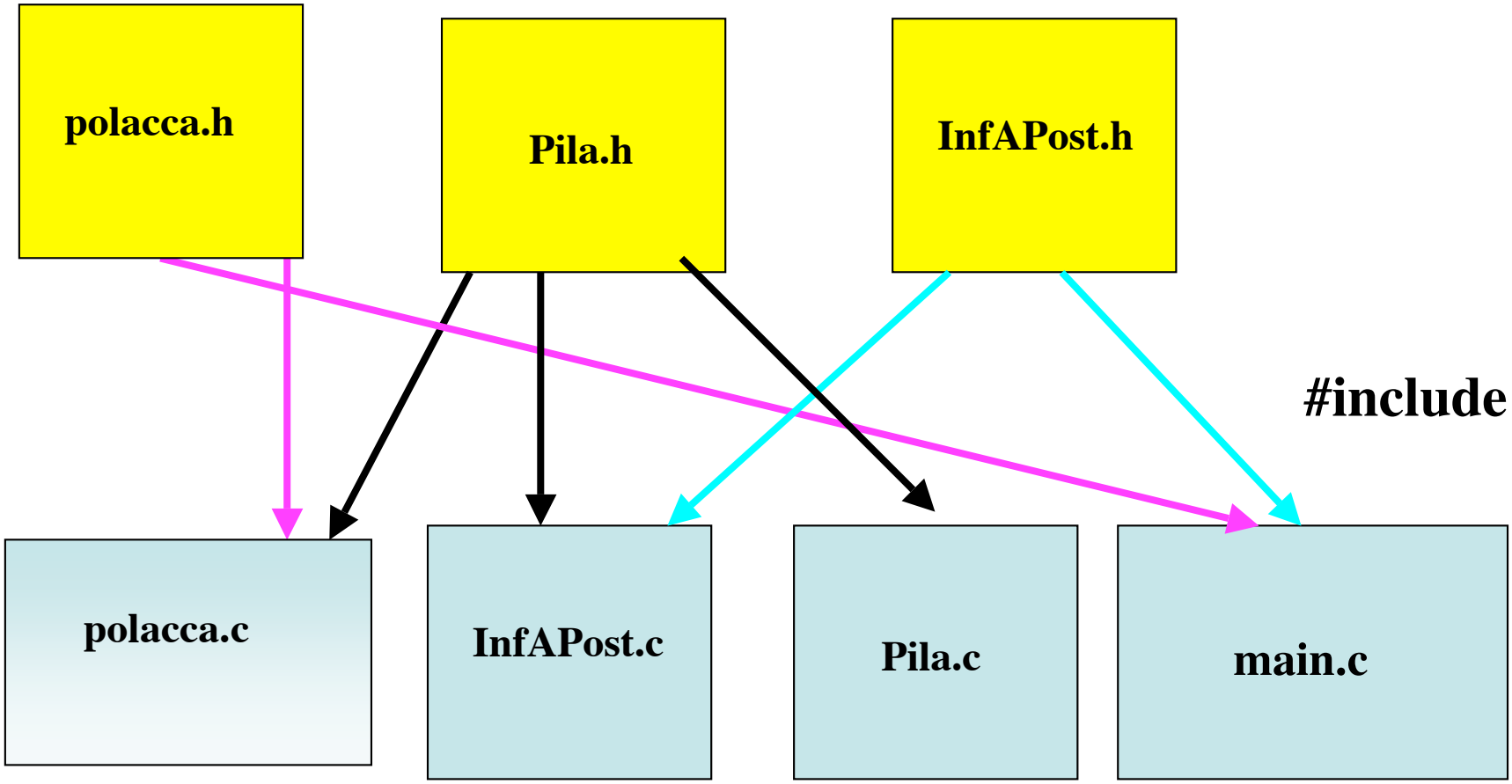
altrimenti impila il simbolo letto.

Svuota la pila concatendando i simboli estratti alla postfissa

Esempio: Infix = 20 + 15 * (100 – 5) – 21

Postfix = 20 15 100 5 - * + 21 -





I files coinvolti nel calcolo e le inclusioni degli header

Da Infisso a Postfisso: il file InfAPost.h

```
void Conversione(char* Infix,char** Postfix);
/*prec: Infix != NULL && Postfix != NULL && *Postfix != NULL &&
**Postifx == '\0' && Infix contiene una'espressione aritmetica
   infissa in cui uno spazio divide operatori e operandi
postc: restituisce in Postfix la stringa in notazione postfissa
   equivalente a quella in Infix*/

int Operando(char ch);
/*postc: restituisce vero se il carattere è un operando di
   un'espressione aritmetica, falso altrimenti*/

int Precedenza(char op1, char op2);
/* postc:  restituisce vero se op1 ha precedenza su op2, falso
   altrimenti */

char * accodaCar(char * s, char c);
/*prec: s != NULL
postc: restituisce la stringa s ottenuta concatenando il carattere c
   a destra a s. */
```

Da Infisso a Postfisso: il file InfAPost.c - le inclusioni

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
#include "InfAPost.h”
#include "pila.h"
```

Da Infisso a Postfisso: il file InfAPost.c

```
void Conversione(char* Infix, char** Postfix)
{PilaP  PILA;
char simbCorr,simbCima;
PILA = costrPila(strlen(Infix));
simbCorr =  *Infix;
while  (simbCorr != '\0')
{if (Operando(simbCorr) || (simbCorr == ' '))
 *Postfix = accodaCar(*Postfix,simbCorr);
 else
   {while ((!vuota(PILA)) && Precedenza(top(PILA),simbCorr))
    {simbCima = pop(PILA);
     *Postfix = accodaCar(*Postfix,simbCima);}
     if ((!vuota(PILA)) && (simbCorr == ')'))
        pop(PILA);   /* eliminiamo la parentesi aperta
   corrispondente alla corrente chiusa */
    else  push(simbCorr,PILA);}
simbCorr = *(++Infix); }
while (!vuota(PILA))
      {simbCima = pop(PILA);
       *Postfix = accodaCar(*Postfix,simbCima);}
}
```

Da Infisso a Postfisso: il file InfAPost.c

```
int Operando(char ch)
/*postc: restituisce vero se il carattere
  è un operando di un'espressione
  aritmetica, falso altrimenti*/
{return
( ((ch >= 'a') && (ch <= 'z')) ||
((ch >= 'A') && (ch <= 'Z')) ||
((ch >= '0') && (ch <= '9')) );}
```

Da Infisso a Postfisso: il file InfAPost.c

```
int Precedenza(char op1, char op2)
/* postc:  restituisce vero se op1 ha precedenza
   su op2 falso altrimenti*/
 {if (op1 == '(')   return  0;
  else
  if (op2 == '(') return 0;
  else
  if (op2 == ')') return 1;
  else
  if ((op1 == '*') || (op1 == '/')) return 1;
  else
  if ((op2 == '*') || (op2 == '/'))
  return 0;
  else return 1; }
```

Da Infisso a Postfisso: il file InfAPost.c

```
char * accodaCar(char * s, char c)
/*prec: s != NULL
postc: restituisce la stringa s ottenuta concatenando il
   carattere c a destra in s. */
{char * temp = s;
assert(s);
while (*s != '\0')  s++;
*s = c;
*(++s) = '\0';
return temp;}
```

il main

```
#include "polacca.h"
#include "InfAPost.h"

int main (void)
{char * str,*strpost;
INSERISCI  IN STR UNA ESPRESSIONE ARITMETICA INFISSA
printf("Espressione:%s \n ",str);
strpost = calloc(strlen(str),sizeof(char));
Conversione(str,&strpost);
printf("L'equivalente postfissa  è %s\n", strpost);
printf("Il valore è %d\n", valutaPolacca(strpost));
return 0;}
```

Nel mail calcoliamo il valore di un’espressione aritmetica infissa passando attraverso la conversione a postfissa.