

# Informatica III – 17 /1/ 12 – info 3 – Nome

## Record di attivazione A

Scrivi una funzione in C che dato in ingresso una stringa restituisce true se la stringa è palindroma. Al solito, scrivi tre versioni: una non ricorsiva, una ricorsiva senza tail recursion e una ricorsiva con tail recursion.

Specifica esattamente i parametri che passi alla procedura, il tipo di passaggio utilizzato e il loro significato.

Scrivi anche un main di esempio in cui chiami la funzione con una stringa (corta) di esempio e invocala in modo assegni il risultato ad una variabile globale. Non usare alcuna altra variabile globale.

Disegna il record di attivazione per tutte e tre le versioni fino alla massima estensione del record di attivazione. Nel caso di tail recursion, spiega quali ottimizzazioni hai adottato o potresti adottare.

## Record di attivazione in C++

Considera questo codice C++:

```
int A = 9;

int foo(int &y) {
    y++;
    if (y > 0 ) {
        int q = 10;
        A = y;
        return g(y);
    } else {
        int y = 90;
        return foo(y);
    }
}

int g(int A) {
    int x = 10;
    return A+ x;
}

int main() {
    int q = -50 + A;
    cout << "1. " << m(q) << endl;
    cout << "2. " << q << endl;
}
```

Quale è l'output? Scrivi il record di attivazione per la chiamata delle due funzioni.

## Cyclone

A cosa servono i puntatori() a char zero term? Come si dichiarano? Come si usano? Fai un piccolo esempio. Quali sono i vantaggi e svantaggi del loro uso?

## Tipi opachi in C

Come si usano i tipi opachi in C? A cosa servono? Fai un piccolo esempio.

## C++

Considera il seguente codice?

```
class Computer {
    private:
        int pri() { return 1; }
    public:
        int pub() { return 2; }
        virtual int vpub() { return 3; }
}
```

```

};
class Smartphone: private Computer {
public:
    int vpub() { return 5; }
};

class Laptop: public Computer {
private:
    int pri() { return 6; }
public:
    int vpub() { return 7; }
};
int main() {
1.    Computer p;
2.    cout << p.pri() << endl;
3.    cout << p.pub()<< endl;
4.    cout << p.vpub()<< endl;

5.    Smartphone s;
6.    cout << s.pri() << endl;
7.    cout << s.pub()<< endl;
8.    cout << s.vpub()<< endl;

9.    Laptop m;
10.   cout << m.pri() << endl;

11.   cout << m.pub()<< endl;
12.   cout << m.vpub()<< endl;

13.   Smartphone * pv = &p;
14.   Laptop * pm = &p;

15.   Computer * pp = &s;
16.   cout << pp->pri()<< endl;
17.   cout << pp->pub()<< endl;
18.   cout << pp->vpub()<< endl;

19.   Computer * pp2 = &m;
20.   cout << pp2->pri()<< endl;
21.   cout << pp2->pub()<< endl;
22.   cout << pp2->vpub()<< endl;

23.   Computer e1 = s;
24.   cout << e1.pri() << endl;
25.   cout << e1.pub()<< endl;
26.   cout << e1.vpub()<< endl;

27.   Computer e2 = m;
28.   cout << e2.pri() << endl;
29.   cout << e2.pub()<< endl;
30.   cout << e2.vpub()<< endl;
}

```

Quale è l'output/effetto prodotto da ogni linea numerata del main? Se contiene un errore scrivi errore, spiega l'errore e ignora la linea (ed eventuali istruzioni che dipendono da essa).

## Dinamic Binding in Java

Date le seguenti dichiarazioni:

```

class Computer{
    public int m(double d){return 5;}
}

class Smartphone extends Computer{
    public int m(double d){return 6;}
    public int m(int d){return 7;}
}

Object oc = new Computer();
Computer cs = new Smartphone();

Smartphone ss = new Smartphone();

```

Quale è il valore delle seguenti tre espressioni spiegando bene (cioè anche il processo di early e late binding dove necessario) il perché (anche se le ritieni errate):

```

oc.equals(oc);
oc.equals(cs);
cs.m(2);

cs.m(2.0);
ss.m(2);
ss.m(2.0);

```

## Java

C'è la covarianza tra tipi generici rispetto al tipo base? Ad esempio se A è sottoclasse di B,  $C\langle A \rangle$  è sottoclasse di  $C\langle B \rangle$ ? Quale è la gerarchia dei tipi e supertipi di  $C\langle A \rangle$ ?

## Testing – 6 crediti

### *Testing programmi*

Dato il seguente programma:

```
foo(int x, int y){
    if( y % 2 == 0 (x > y || x > 5)){
        x--;
    }
}
```

Disegna il grafo del programma e trova i casi di test per la copertura delle istruzioni, delle decisioni, delle condizioni e l'MCDC

### Testing FSM

Scrivi una macchina a stati finiti con almeno 3 stati e trovanne un transition tuor (anche non euleriano). Introduci un difetto e scopri se il tuo test è in grado di scoprirlo.

## KeyHoare – 17 /1/ 12 – info 3 – Nome \_\_\_\_\_

---

Istruzioni:

- scarica lo zip dalla directory indicata dal professore e unzippala sul desktop
- per farlo partire, doppio click su startKeyHoare.bat
- carica il tuo file (se hai errore chiudi, correggi e riapri)
- salva la prova con File -> Save .proof.

### Problema A:

Considera il programma che dovrebbe calcolare  $n \cdot x$

```
z = 0;
if (n>0) {
    an = n;
} else{
    an = -n;
}
while (an>0) {
    z = z + x;
    an = an-1;
}
if (n < 0) {z = -z;}
else{}
```

Scrivi le pre e post condizioni e dimostra la correttezza.

Aiuto: per rimuovere istruzioni vuote usa removeBlock.

---

## KeyHoare – 17 /1/ 12 – info 3 – Nome \_\_\_\_\_

---

Istruzioni:

- scarica lo zip dalla directory indicata dal professore e unzippala sul desktop
- per farlo partire, doppio click su startKeyHoare.bat
- carica il tuo file (se hai errore chiudi, correggi e riapri)
- salva la prova con File -> Save .proof.

### Problema B:

Considera il programma che dovrebbe calcolare  $n \cdot a$

```
y = 0;
if (n>0) {
    an = n;
} else{
    an = -n;
}
while (an>0) {
    y = y + a;
    an = an-1;
}
if (n < 0) {y = -y;}
else{}
```

Scrivi le pre e post condizioni e dimostra la correttezza.

Aiuto: per rimuovere istruzioni vuote usa removeBlock.

