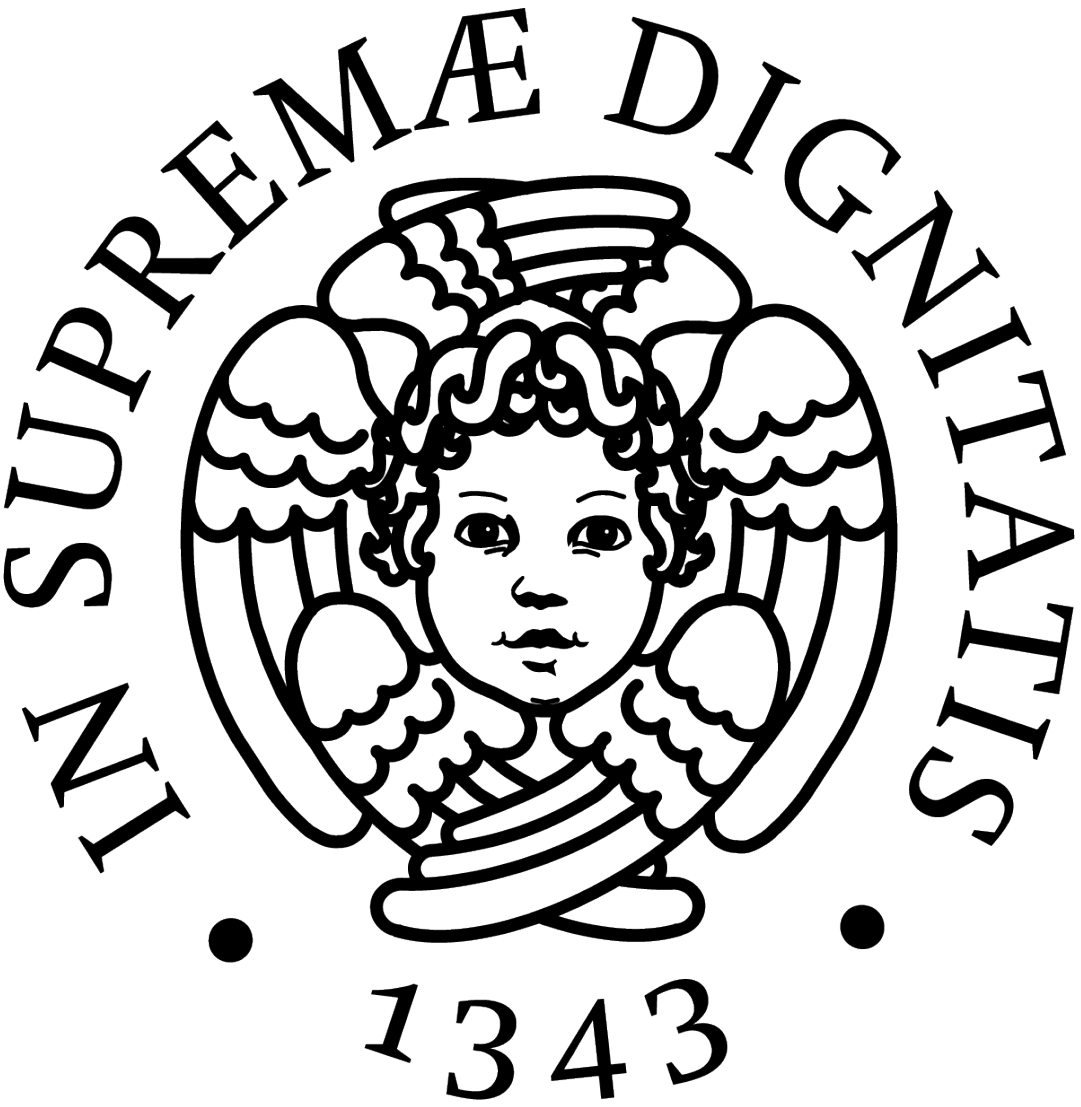
Università di Pisa



Dipartimento di Ingegneria dell'Informazione

CdL Ingegneria Informatica

Briciole di Fondamenti di Programmazione

Autore:
Francesco Zollo

Liste

Struct Lista:

```
struct elem{
   int inf;
   elem* pun;
};
```

Stampa Lista:

```
void stampalista(lista p0){
 elem* p=p0;
 while (p!=0){
    cout << p->inf << ' ';
    p=p->pun;
 }
}
```

Inserimento in Testa:

```
void instesta(elem* &p0, int a){
   elem* p = new elem;
   p->inf=a;
   p->pun=p0;
   p0=p;
}
```

Estrazione dalla Testa:

```
bool esttesta(elem* &p0, int &a){
   elem* p=p0;
   if(p0==0) return false;
   a=p0->inf;
   p0=p0->pun;
   delete p;
   return true;
}
```

Inserimento in Fondo:

```
void insfondo(elem* &p0, int a){
   elem *p;
   elem* q;
   for(q=p0; q!=0; q=q->pun) p = q;
   q=new elem;
   q->inf=a;
   q->pun=0;
   if(p0==0) p0=q;
   else p->pun=q;
}
```

Estrazione dal fondo:

```
bool estfondo(elem* &p0, int &a){
   elem *p=0;
   elem *q;
   if(p0==0) return false;
   for(q=p0; q->pun!=0; q=q->pun) p=q;
   a=q->inf;
// controlla se si estrae il primo elemento
   if (q==p0) p0=0;
   else p->pun=0;
   delete q;
   return true;
}
```

Inserimento Ordinato:

```
void inserimento(elem* &p0, int a){
   elem* p = 0;
   elem* q;
   elem* r;
   for(q=p0; q!=0 && q->inf<a; q=q->pun) p=q;
   r=new elem;
   r->inf=a;
   r->pun=q;
// controlla se si deve inserire in testa
   if (q==p0) p0=r;
   else p->pun=r;
}
```

Estrazione di un Elemento:

```
bool estrazione(elem* &p0, int a){
   elem* p=0;
   elem* q;
   for(q=p0; q!=0 && q->inf!=a; q=q->pun) p=q;
   if(q==0) return false;
   if(q==p0) p0=q->pun;
   else p->pun = q->pun;
   delete q;
   return true;
}
```

Estrazione Elemento Ordinato:

```
bool estrazione_ordinata(elem* &p0, int a){
   elem* p=0;
   elem* q;
   for(q=p0; q!=0 && q->inf<a; q=q->pun) p=q;
   if((q==0) || (q->info>a)) return false;
   if (q==p0) p0=q->pun;
   else p->pun=q->pun;
   delete q;
   return true;
}
```

Pile

Generalità: Struttura LIFO: Last In First Out

Struct Pila:

```
const int DIM = 5;
struct pila{
 int top;
 int stack[DIM];
};
```

Inizializzazione Pila:

```
void inip(pila& pp){
   pp.top=-1;
}
```

Pila Vuota?

```
bool empty(const pila& pp){
   if(pp.top==-1) return true;
   return false;
}
```

Pila Piena?

```
bool full(const pila& pp){
   if(pp.top==DIM-1) return true;
   return false;
}
```

Inserimento in Pila:

```
bool push(pila& pp, int s){
   if(full(pp)) return false;
   pp.stack[++(pp.top)]=s;
   return true;
}
```

Estrazione dalla Pila:

```
bool pop(pila& pp, int &s){
   if(empty(pp)) return false;
   s=pp.stack[(pp.top)--];
   return true;
}
```

Stampa Pila:

```
void stampa(const pila &pp){
 cout << "Elementi contenuti nella pila: " << endl;
 for(int i = pp.top; i >= 0; i--)
 cout << '[' << i << "] " << pp.stack[i] << endl;
}
```

Code

Generalità: Struttura FIFO: First In First Out
front: dove avviene l'estrazione
back: dove avviene l'inserimento
posso inserire fino a DIM-1

Struct Coda:

```
const int DIM=5;
struct coda{
   int front;
   int back;
   int queue[DIM];
};
```

Inizializzazione Coda:

```
void inic(coda& cc){
   cc.front=0;
   cc.back=0;
}
```

Coda Vuota?

```
bool empty(const coda& cc){
   if(cc.front==cc.back) return true;
   return false;
}
```

Coda Piena?

```
bool full(const coda& cc){
   if(cc.front==(cc.back+1)%DIM) return true;
   return false;
}
```

Inserimento:

```
bool insqueue(coda& cc, int s){
   if(full(cc)) return false;
   cc.queue[cc.back]=s;
   cc.back=(cc.back+1)%DIM;
   return true;
}
```

Estrazione:

```
bool esqueue(coda& cc, int &s){
   if(empty(cc)) return false;
   s=cc.queue[cc.front];
   cc.front=(cc.front + 1)%DIM;
   return true;
}
```

Stampa Coda:

```
void stampa(const coda& cc){
for(int i=cc.front; i%DIM!=cc.back; i++)
   cout << cc.queue[i%DIM] << endl;
}
```

Libreria <csting>

- `char *strcpy(char *dest, const char *sorg );`
  Copia *sorg* in *dest* compreso il carattere nullo e restituisce *dest*.
- `char *strcat(char *dest, const char *sorg );`
  Mette in coda *sorg* a *dest*, e restituisce *dest*. Il carattere nullo è alla fine di tutto.
- `int strlen(const char *string);`
  Restituisce la lunghezza senza il carattere nullo.
- `int strcmp(const char *s1, const char *s2 );`
  Confronta alfabeticamente $s1$,$s2$: $\begin{cases} - \rightarrow \text{s1<s2} \\ 0 \rightarrow \text{s1=s2} \\ + \rightarrow \text{si>s2} \end{cases}$
- `char *strchr(const char *string, char c );`
  Restituisce un puntatore alla posizione dove si trova *c* in *string*, e 0 se non c'è.

Ordinamento di Vettori

Scambia (propedeutica):

```
void scambia(int vettore[], int x, int y){
   int lavoro=vettore[x];
   vettore[x]=vettore[y];
   vettore[y]=lavoro;
}
```

Selection-Sort:

```
void selectionSort(int vettore[], int n){
int min;
for(int i=0; i<n-1; i++){
   min = i;
   for(int j=i+1; j<n; j++)
        if(vettore[j]<vettore[min]) min=j;
   scambia(vettore,i,min);
   }}
```

Bubble-Sort:

```
void bubble(int vettore[], int n){
    for(int i=0 ; i<n-1; i++)
        for (int j=n-1; j>i; j--)
            if(vettore[j]<vettore[j-1])
                 scambia(vettore, j, j-1);
}
```

Ricerca in Vettori

Ricerca Lineare:

```
bool ri(int v[],int inf,int sup,int k,int &pos){
   bool trovato=false;
   while((!trovato) && (inf<=sup)){
      if(v[inf]==k){
         pos=inf;
         trovato=true;
      }
   inf++;
   }
   return trovato;
}
```

Ricerca Binaria (vettore ordinato):

```
bool rib(int ordv[],int inf,int sup,int k,int &pos){
   while(inf<=sup){
      int medio=(inf+sup)/2;
      if(k>ordv[medio]) inf=medio+1;
      else if(k<ordv[medio]) sup=medio-1;
      else{
           pos = medio;
           return true;
          }
   }
   return false;
}
```

Classi

Costruttori:

-file .h

```
nomeclasse(type n1,..,type nn);
```

-file .cpp

```
nomeclasse::nomeclasse(type n1,..,type nn){...}
```

-main

```
nomeclasse nome(1,...,n);
```

Distruttori:

-file .h

```
~nomeclasse();
```

-file .cpp

```
nomeclasse::~nomeclasse(){...}
```

Costruttori di Copia:

-file .h

```
nomeclasse(const nomeclasse&);
```

-file .cpp

```
nomeclasse::nomeclasse(const nomeclasse& m){...}
```

-main

```
nomeclasse m1(m);
```

Operatori di Conversione:

-file .h

```
operator type()const;
```

-file .cpp

```
nomeclasse::operator type()const{...}
```

-main

```
type(nomeclasse);
```

Operatori di Incremento Prefisso:

-file .h

```
nomeclasse& operator++();
```

-file .cpp

```
nomeclasse& nomeclasse::operator++(){...}
```

-main

```
++nomeclasse;
```

-globale

```
friend nomeclasse& operator++(nomeclasse&);
```

Operatori di Incremento Postfisso:

-file .h

```
nomeclasse operator++(int);
```

-file .cpp

```
nomeclasse nomeclasse::operator++(int){...}
```

-main

```
nomeclasse++;
```

-globale

```
friend nomeclasse& operator++(nomeclasse&,int);
```

Operatori di Uscita:

-file .h

```
friend ostream& operator<<(ostream&,const classe&);
```

-file .cpp

```
ostream& operator<<(ostream& os, const classe& m){...}
```

-main

```
cout << classe;
```

Rappresentazione

Numeri Interi:

Modulo e Segno

- a $\Rightarrow$ A: $A = a_{p-1}...a_0 = (sgn(a), ABS(a))$
- A $\Rightarrow$ a: $a = (a_{p-1} == 0)? + ABS(a) : -ABS(a)$
- Intervallo: $[-(2^{p-1}-1), 2^{p-1}+1]$

Complemento a Due

- a $\Rightarrow$ A: $A = a_{p-1}...a_0 = (a >= 0)?ABS(a) : (2^p - ABS(a))$
- A $\Rightarrow$ a: $a = (a_{p-1} == 0)? + A : -(2^p - A)$
- Intervallo: $[-2^{p-1}, +2^{p-1}-1]$

BIAS

- a $\Rightarrow$ A: $A = a_{p-1}...a_0 = a + (2^{p-1}-1)$
- A $\Rightarrow$ a: $a = A - (2^{p-1}-1)$
- Intervallo: $= [-BIAS, BIAS+1]$
  $[-(2^{p-1}-1), (2^{p-1}-1)+1]$

Numeri Reali:

Virgola Fissa

- R $\Rightarrow$ r:
  f bit $\rightarrow$ parte frazionaria
  (f-p) bit $\rightarrow$ parte intera
  $R = a_{p-f-1}...a_0...a_{-f}$
  $r = \sum_{i=-f}^{p-f-1} a_i \beta^i$
  $= \underbrace{a_{p-f-1}\beta^{p-f-1} + ... + a_0\beta^0}_{parte\ intera} + \underbrace{a_{-1}\beta^{-1} + ... + a_{-f}\beta^{-f}}_{parte\ frazionaria}$
- r $\Rightarrow$ R: se $f_0 \neq 0$
  $f_{-1} = F(2 * f_0)$ $\quad a_{-1} = I(2 * f_0)$
  $f_{-2} = F(2 * f_{-1})$ $\quad a_{-2} = I(2 * f_{-1})$
  $\Downarrow$
  $f_{-j} = 0 \Rightarrow$ STOP!

Virgola Mobile

- r $\Rightarrow$ R=<s,E,F>

  S=codifica segno , E=codifica esponente su K bit
  F=codifica parte frazionaria su G bit
  $r = (s == 0)?[(1+f)2^e] : [-(1+f)2^e]$
  $f = \frac{F}{2^G}$, $m = 1 + f$, $e = E - (2^{k-1}-1)$
- R=<s,E,F> $\Rightarrow$ r

  $F = f2^G$, $f = 1 - m$, $E = e + (2^{k-1}-1)$