RICAPITOLAZIONE - Esercizio: liste di punti colorati

Dato un file PUNTI.TXT, di punti colorati, eliminare quelli piu' a destra e salvare in un nuovo file

0) lista, nomeFile, nomeFile2, puntMaxAscissa,
1) lettura nomeFile e costr. lista:
   `lista = costrListaDaFile(nomeFile)`
2) se lista VUOTA,
   `FINE else stampaLista(lista)`
3) trovare ascissa massima in lista
   `puntMaxAscissa = pMaxAsc(lista)`
4) elimina dalla lista i punti di ascissa massima
   `eliminaTutti(&lista, puntMaxAscissa->info.asc)`
5) stampa lista
6) leggi nomeFile2 e
   `stampaListaSuFile(lista, nomeFile2);`
7) eseguire `deallocaLista(&lista);`
8) stampa lista (per verifica)

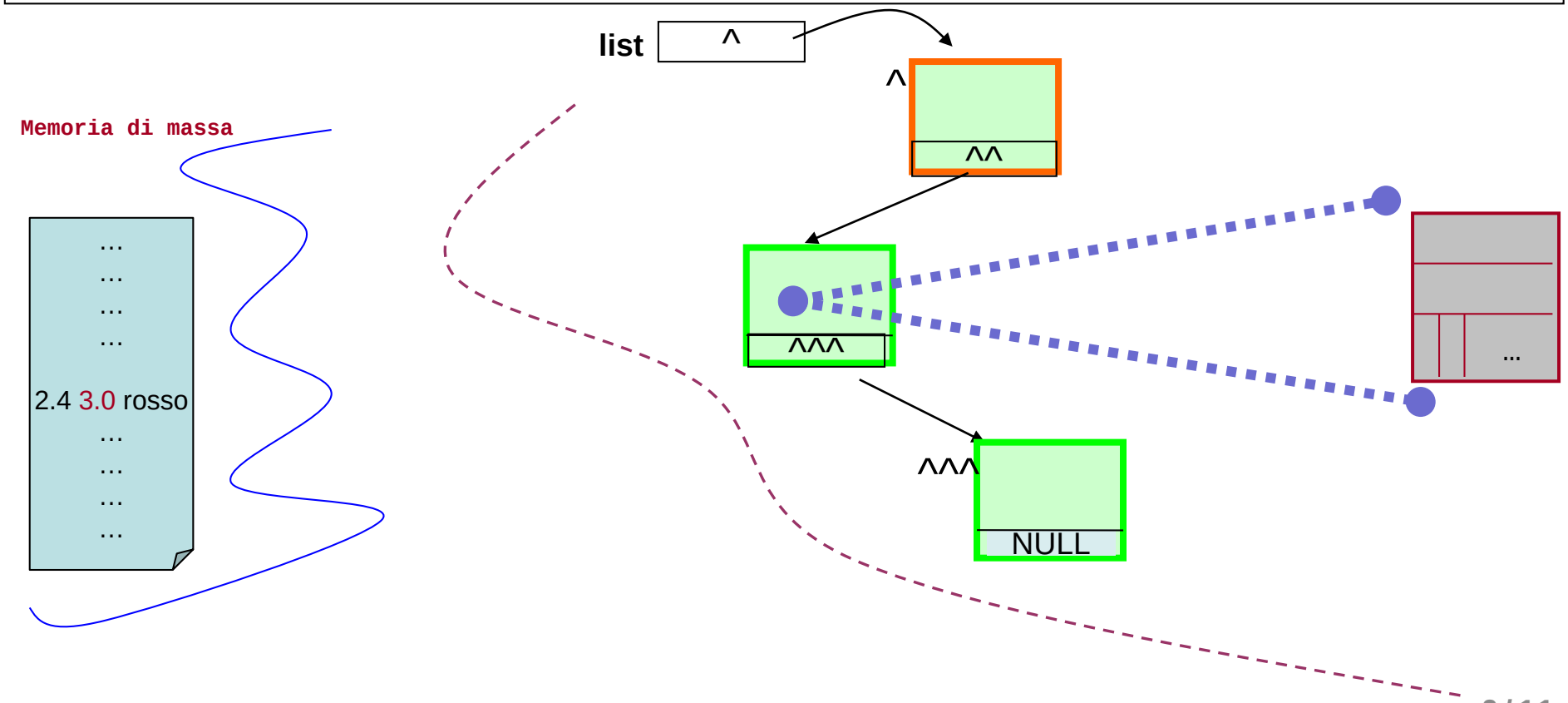
NB puntatore ad un nodo di massima ascissa

NB un valore preso da un nodo che verra' eliminato

NB stampe di controllo

liste di punti colorati: strutture dati e main()

```
definizione  TipoPunto
                        /*i campi info dei nodi sono di tipo TipoPunto */
                        typedef TipoPunto TipoElem
definizione      TipoNodo (struct{TipoElem info; ... * next})
                 TipoLista   (TipoNodo *)
                 PuntNodoLista   (TipoNodo *)
```



liste di punti colorati: strutture dati e main()

```
definizione TipoPunto
                    /*i campi info dei nodi sono di tipo TipoPunto */
                    typedef TipoPunto TipoElem
definizione     TipoNodo (struct{TipoElem info; ... * next})
                TipoLista   (TipoNodo *)
                PuntNodoLista   (TipoNodo *)
```

```
int main() {
  TipoLista lista;
  char nomeFile[20], nomeFile2[20];
  PuntNodoLista puntMaxAscissa, aux;
...
/* lettura nomeFile */
...
  lista = costrListaDaFile(nomeFile);
  if (lista==NULL) { messaggio
    return 0;
  }
stampaLista (lista);                    ..........
```

liste di punti colorati: main()

```
int main() {
  TipoLista lista;
  char nomeFile[20], nomeFIle2[20];
  PuntNodoLista puntMaxAscissa, aux;
...
/* lettura nomeFile */
...
  lista = costrListaDaFile(nomeFile);
  if (lista==NULL) { messaggio
    return 0;
  }
  stampaLista (lista);
  puntMaxAscissa = pMaxAsc(lista);
  eliminaTutti (&lista, puntMaxAscissa->info.asc);
  stampaLista (lista);
/* lettura nomeFile2 */ ...
  stampaListaSuFile (lista, nomeFile2);
  deallocaLista(&lista);
  stampaLista(lista);
return 0;
}
```

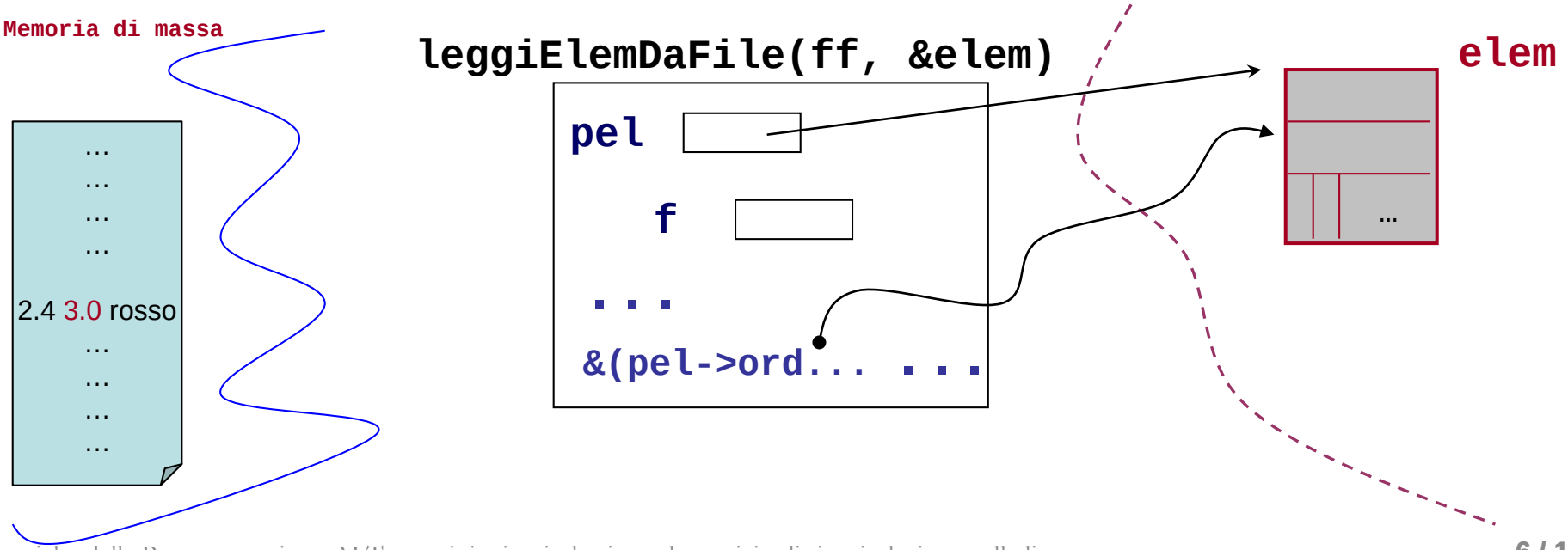
costruzione lista da file

```
TipoLista costruisciListaDaFile ( char * nmf) {
  FILE * fin;
  TipoElem elem;
  TipoLista ris = NULL;

  fin = fopen(nmf, "r");
  if (fin==NULL) {
    printf ("\n problemi in apertura file dati !!\n");
    return NULL;
  }
                  /* FORMATO FILE: 1 punto per linea */
  while (!feof(fin)) {
    leggiElemDaFile(fin, &elem);
    insTestaLista(&ris, elem);      /* o in coda, o ord ... */
  }
return ris;
}
```

alcune funzioni accessorie

```
void insTestaLista ( TipoLista *plis, TipoElem el) { ...
```

```
void leggiElemDaFile (FILE * f, TipoElem *pel) {
  /* leggiPuntoDAFIle (f, pel) ... */
  fscanf(f, "%lf %lf %s\n",
            &(pel->ascissa), &(pel->ord), pel->colore);
return;
}
```



```
void insTestaLista ( TipoLista *plis, TipoElem el) { ...
```

```
void leggiElemDaFile (FILE * f, TipoElem *pel) {
  /* leggiPuntoDAFIle (f, pel) ... */
  fscanf(f, "%lf %lf %s\n",
            &(pel->ascissa), &(pel->ord), pel->colore);
return;
}
```

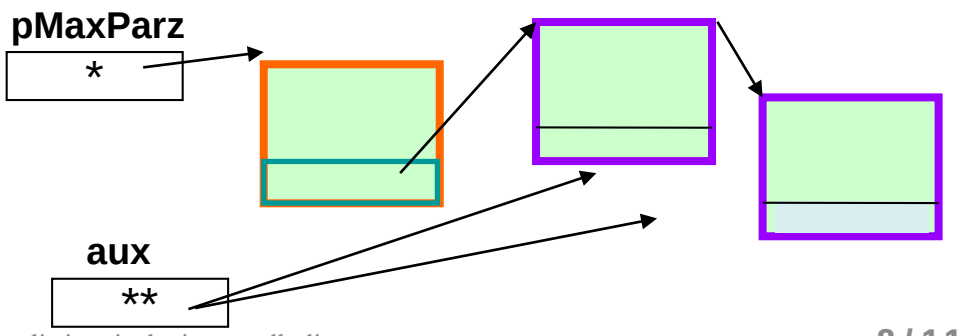
```
void stampaLista (TipoLista l) {
  while (l) {
    stampaElem(l->info);
    l = l->next;
  }
return;
}
```

```
void stampaElem (TipoElem el) {
  /* stampaPunto (el) ... */
  printf ("(%g, %g) %s\n",
    el.asc, el.ord, el.colore);
return;
}
```

Calcolo dell'ascissa massima nella lista di punti

Se la lista non e' vuota ha almeno un nodo (massimo parziale iniziale);
poi si confronta il massimo parziale con i nodi della lista, a partire dal secondo;

```
/* restituisce il puntatore ad un nodo con massima ascissa (o NULL se l vuota) */
TipoNodo * pMaxAsc (TipoLista l) {
                /* PuntNodoLista o TipoNodo  *, e' lo stesso */
  TipoNodo * pMaxParz,                    /* massimo parziale ascisse */
           * aux;
  pMaxParz = l;                           /* inizializzazione */
  if (l) {
    aux = l->next;
    while (aux) {
      if (pMaxParz->info.asc < aux->info.asc)
         pMaxParz = aux;                  /* nuovo massimo parziale */

      aux = aux->next;
    }
  }
return pMaxParz;
}
```



Eliminazione dei punti con ascissa massima

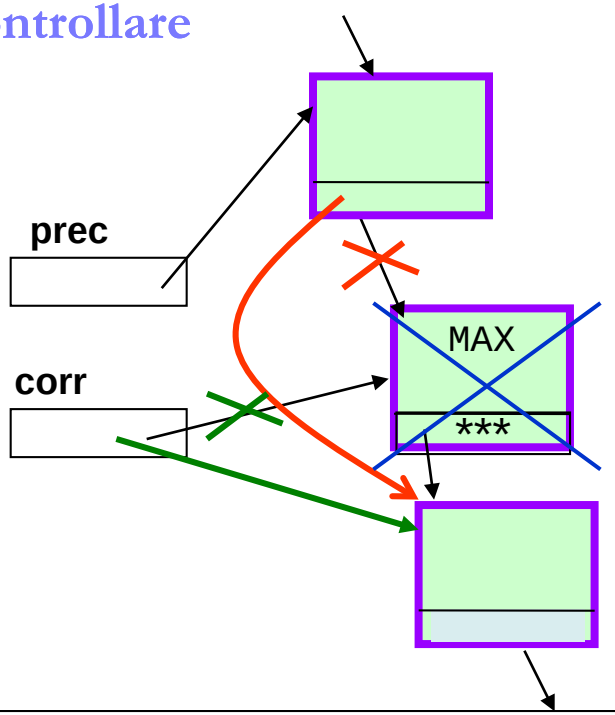
Schema di eliminazione con record generatore e doppio puntatore: `prec/corr`
Si tratta di eliminare quei nodi (`*corr`) per cui `corr->info.asc >= ascMax`;
Bisogna tener conto del fatto che dopo una eliminazione il ciclo di ricerca continua per eliminare altri eventuali nodi; per cui `corr` va riposizionato a dovere

```
/* elimina da l tutti i nodi aventi ascissa maggiore o uguale ascMax */
void eliminaTutti (TipoLista *plis, double ascMax) {
  TipoNodo * pgen, *prec, *corr;

  prec = pgen =malloc(sizeof(TipoNodo));
  if (!pgen) {printf("...acc..."); return; }

  corr = pgen->next = *plis;

  while (corr)
    if (corr->info.asc >= ascMax) {        /* nodo da eliminare */
       /* eliminazione *corr e preparazione prossima iterazione */A)   }
    else {    /* preparazione  "regolare" prossima scansione */B)   }

*plis = pgen->next;                  /* chiusura alg con RG */
free(pgen);
return; }
```

Eliminazione ...: casi A) e B)

A) E' il caso in cui il nodo puntato da corr (*corr) viene eliminato: bisogna
- riposizionare il campo next del nodo precedente a quello daelim
- eliminare il nodo puntato da corr
- posizionare corr in modo che *corr continui ad essere il successore di *prec, cioe' in modo che corr punti al prossimo nodo da controllare

```
prec->next = corr->next;  /* ... */
free ( corr);             /* ... */
corr = prec->next;        /* ... */
```



B) E' il caso in cui dobbiamo solo spostare prec e corr per il controllo sul pross. nodo

```
prec = corr;
corr = corr->next;
```

Stampa lista su file

Va deciso il formato del file, coerentemente con le operazioni di lettura:
come anticipato decidiamo di memorizzare nel file un punto per ogni linea:
`ascissa ordinata colore` (dati separati da un singolo spazio)

```
void stampaListaSuFile (TipoLista l, char nmf[20]) {
  FILE * fout; TipoNodo * aux;
  fout = fopen(nmf, "w");
  if (fout==NULL) {
    printf ("\n problemi in apertura file dati !!\n");
    return;
  }
  aux = l;                    /* inizializzazione scorrimento della lista */
  while (aux) {
    stampaElemSuFile(fout, aux->info);
    aux = aux->next;
  }

return;
}
```

```
void stampaElemSuFile (FILE * f, TipoElem el) {
  /* stampaPunto su File */
  fprinf(f, "%g %g %s\n", el.asc, el.ord, el.colore);
return;
}
```