UNIVERSITY OF TRENTO - Italy

Esercitazione 15

DISI – aa 2024/2025

Pierluigi Roberti
Carmelo Ferrante

Università degli Studi di Trento
pierluigi.roberti@unitn.it

CPP

- Utilizzare progetti di tipo

CPP / C++

- Include e namespace

```
#include <iostream>

using namespace std;

//per usare cout (oggetto libreria standard) si deve invocare
// std::cout
//usando namespace std si può invocare
// cout
// cioè si può omettere std:: prima di cout
```

Allocazione dinamica

- C
- void* malloc(size_t size);
- void* free(void *ptr);

```
// array di 100 interi
int* v;
v=(int*)malloc(sizeof(int)*100);
...
free(v);

// array di 20 caratteri
char* str;
str=(char*)malloc(sizeof(char)*20);
...
free(str);
```

- C++
- new
- delete

```
// array di 100 interi
int* v;
v = new int[100];
...
delete [] v;

// array di 20 caratteri
char* str;
str = new char[20];
...
delete [] str;
```

Allocazione dinamica array

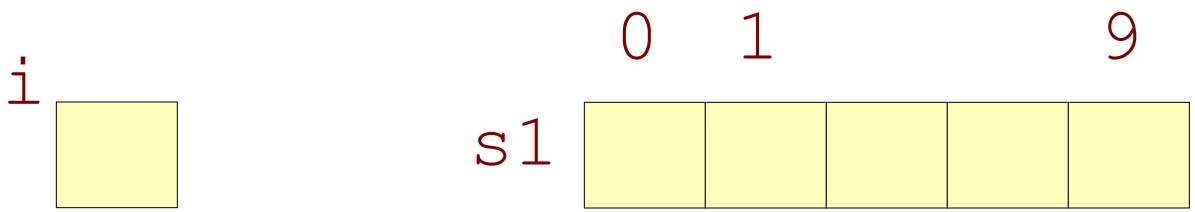
- In genere non genera errori o warning

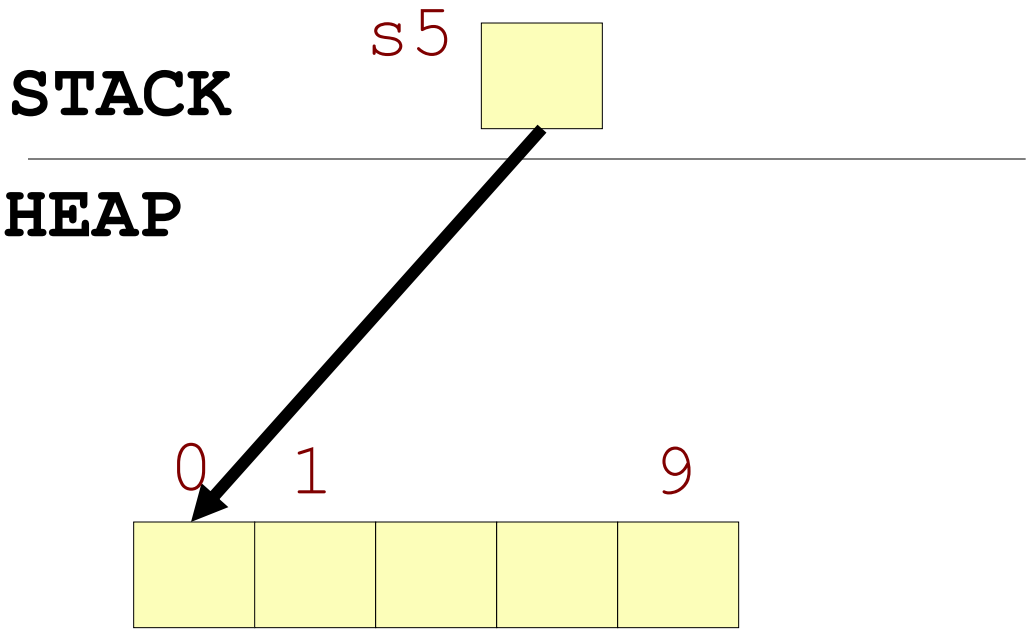
```
// array di 100 interi
int* v;
v = new int[100];
delete v;

// array di 20 caratteri
char* s;
s = new char[20];
delete s;

// array di 20 struct
Tdato* k;
k = new Tdato[20];
delete k;
```

- Formalmente corretta

```
// array di 100 interi
int* v;
v = new int[100];
delete[] v;

// array di 20 caratteri
char* s;
s = new char[20];
delete[] s;

// array di 20 struct
Tdato* k;
k = new Tdato[20];
delete[] k;
```

Allocazione della memoria






```
int main(){
   int i;
   char s1[10];
   char* s5;
   s5=new char[10];
   // . . .
   delete [] s5;
}
```

Esempio 1

```
int main (int argc, char * const argv[]) {
   int v1[10];
   int* v2;
   v2 = new int[10];
   printf("size of int: %d\n",(int)sizeof(int));
   printf("size of char: %d\n",(int)sizeof(char));
   printf("size of int*: %d\n",(int)sizeof(int*));
   printf("size of char*: %d\n",(int)sizeof(char*));

   printf("\nv1: int v1[10]\n");
   printf("v2: int *v2 + alloc dinamica new int[10]\n");
   printf("size of v1: %d\n",(int)sizeof(v1));
   printf("size of v2: %d\n",(int)sizeof(v2));
   delete [] v2;
}
```

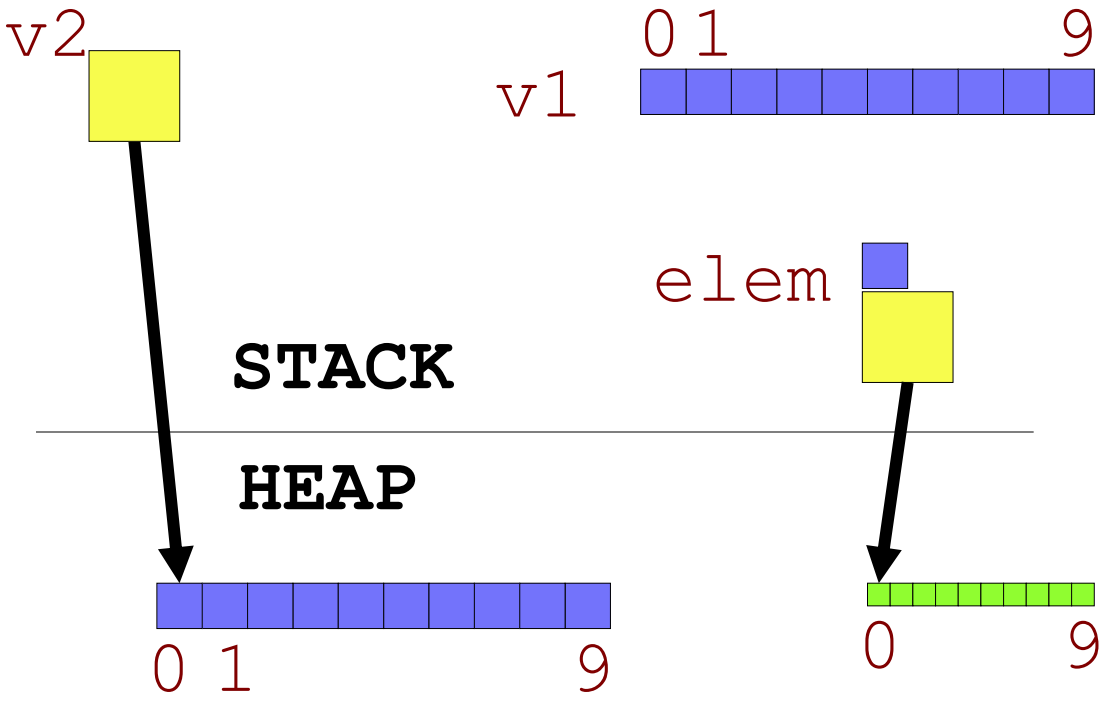
Le dimensioni sono funzione anche dell'architettura del sistema

Esempio 1

```
typedef struct Tdato{
  int i;
  char* s;
} Tdato;

int main (int argc, char * const argv[]) {
   Tdato elem;
   elem.s = new char[10];
   printf("\nelem.s="); scanf("%s",elem.s); //gets(elem.s);

   printf("\nTdato: struct{ int i; char* s; }\n");
   printf("elem: var di tipo Tdato\n");
   printf("size of Tdato: %d\n",(int)sizeof(Tdato));
   printf("size of elem: %d\n",(int)sizeof(elem));
   printf("size of elem.i: %d\n",(int)sizeof(elem.i));
   printf("size of elem.s: %d\n",(int)sizeof(elem.s));

   delete[] elem.s;

   return 0;
}
```

Le dimensioni sono funzione anche dell'architettura del sistema

Allocazione memoria (Esempio 1)



Sizeof

v1 → 40

V2 → 8

elem → 16

elem.i → 4

elem.s → 8

int → 4

int* → 8

char → 1

char* → 8

Le dimensioni sono funzione anche dell'architettura del sistema

4+8 != 16

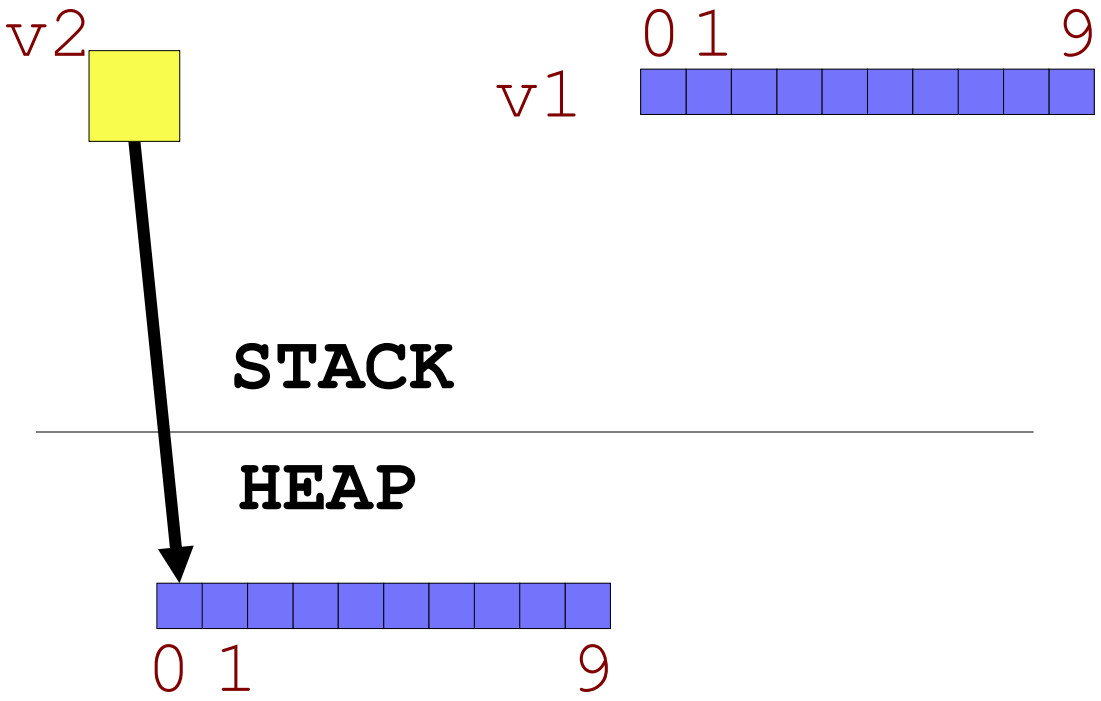
Esercizio 1

Scrivere un programma che:

- Alloca un array v1 di **interi** di dimensione 10
  - Allocazione static (memoria stack)
- Alloca un array v2 di **interi** di dimensione 10
  - Allocazione dinamica (memoria heap)
- Per la dimensione dichiarare una etichetta costante N
- Inizializzare v1 e v2 con valori casuali tra 1 e 9
- Stampare a video il contenuto di v1 e v2

```
Librerie da usare oltre iostream (dipendono dall'IDE)
#include <ctime>      //per il srand(time(0))
#include <random>
#include <cstdlib>    //per il rand()
#include <stdlib.h>   //per il rand()
```

Allocazione memoria (Esercizio 1)



Sizeof

v1 → 40

v2 → 8

v1[i] → 4

v2[i] → 4

Le dimensioni sono funzione anche dell'architettura del sistema

Esercizio 1

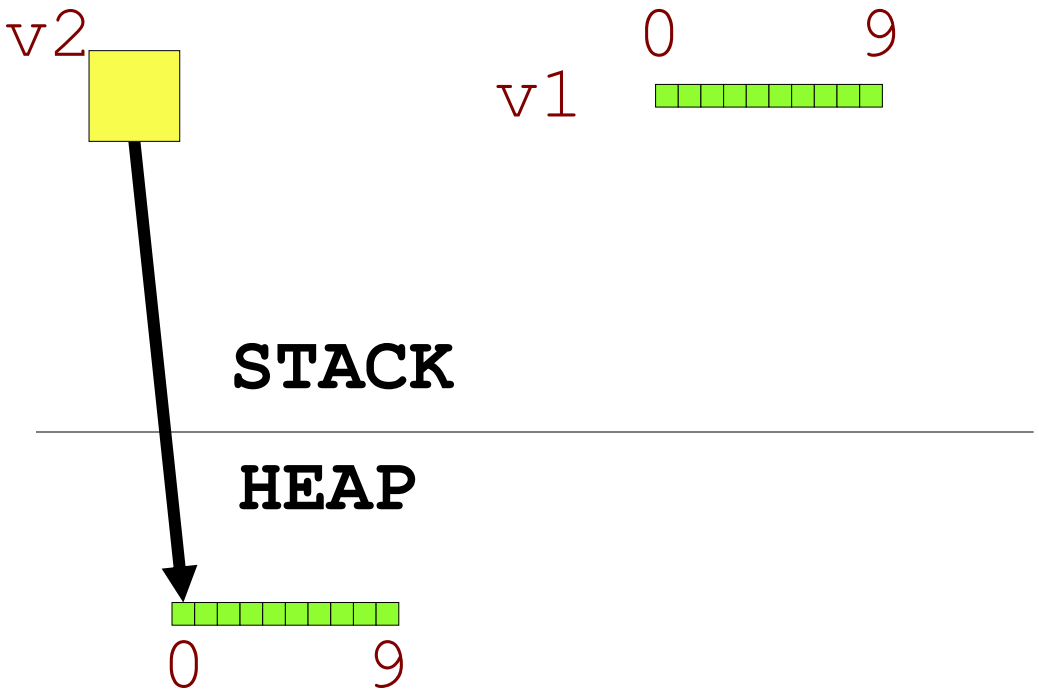
```
int *v2;
v2 = new int[10];

for(int i=0; i<10 ; i++){
   *( v2+i ) = 0;
   //*( v2+i )  →  v2[i]
}
for(int i=0; i<10 ; i++){
   printf("[%d]",v2[i]);
}
printf("\n");

delete[] v2;
```

Esercizio 2

Scrivere un programma che:

- Alloca un array v1 di **caratteri** di dimensione 10
  - Allocazione static (memoria stack)
- Alloca un array v2 di **caratteri** di dimensione 10
  - Allocazione dinamica (memoria heap)
- Per la dimensione dichiarare una etichetta costante N
- Inizializzare v1 e v2 con valori casuali tra ‘a’ e ‘z’
- Stampare a video il contenuto di v1 e v2

Allocazione memoria (Esercizio 2)



Sizeof

v1 → 10

v2 → 8

v1[i] → 1

v2[i] → 1

Le dimensioni sono funzione anche dell'architettura del sistema

Esercizio 2

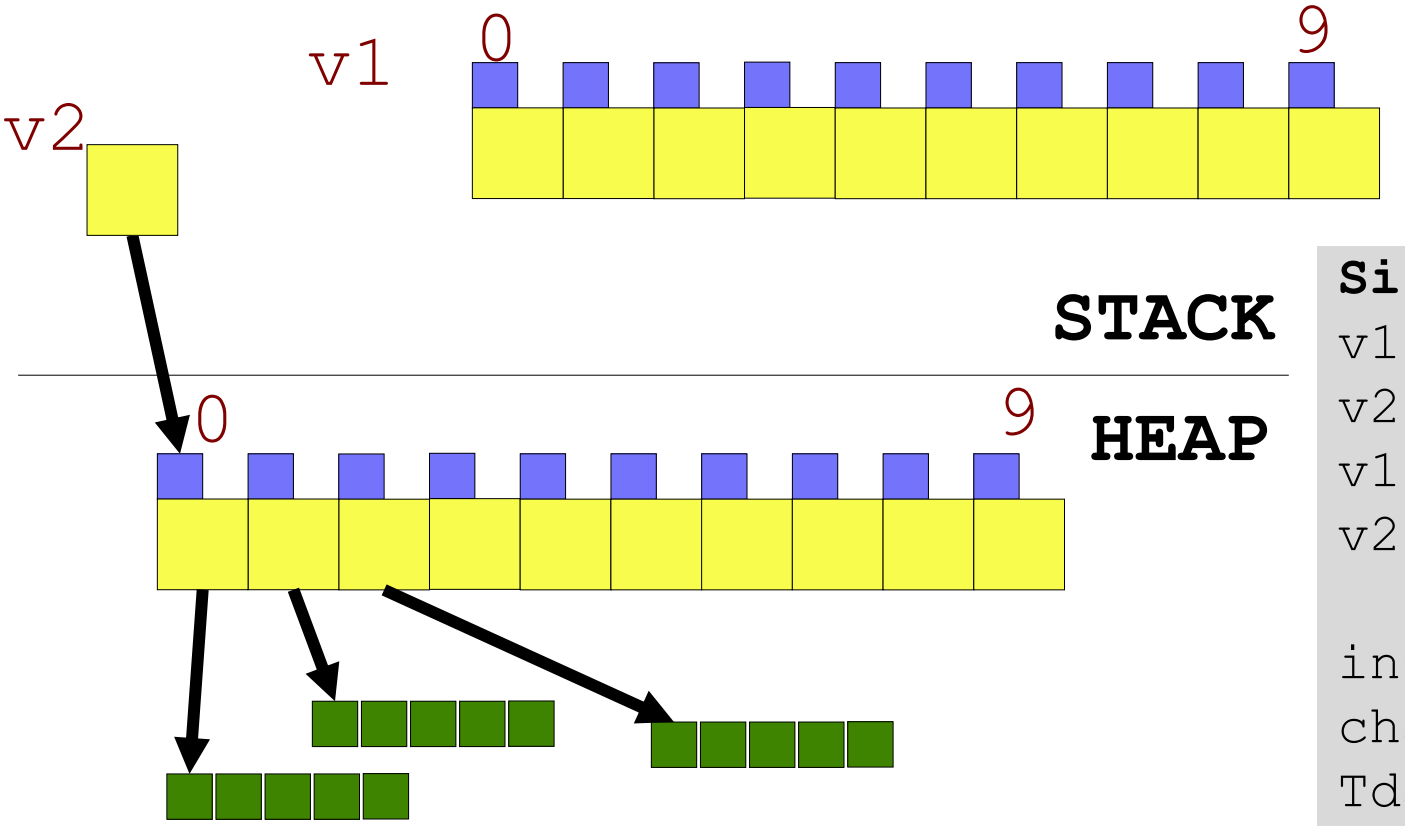
```
char *v2;
v2 = new char[10];
for(int i=0; i<9 ; i++){
   *( v2+i ) = 'a'+i;
   //*( v2+i )  →  v2[i]
}
v2[9]='\0';      //ultimo valore=terminatore
printf("%s",v2);//cout << v2;

delete[] v2;
```

Esercizio 3

Scrivere un programma che:

- Alloca un array v1 di strutture **Tdato** di dimensione 10
  - Allocazione static (memoria stack)
- Alloca un array v2 di strutture **Tdato** di dimensione 10
  - Allocazione dinamica (memoria heap)
- Per la dimensione dichiarare una etichetta costante N
- Per ogni elementi di **Tdato** allocare un array di 5 char
- Inizializzare v1 e v2 con valori casuali
- Stampare a video il contentuto di v1 e v2
- Il tipo di dato strutturato **Tdato** contiene:
  - `int dato;`
  - `char *c;`

Allocazione memoria (Esercizio 3)



Sizeof

```
v1 → 160
v2 → 8
v1[i] → 16
v2[i] → 16

int → 4
char* → 8
Tdato → 16
```

Le dimensioni sono funzione anche dell'architettura del sistema

[int] + [char*] != [Tdato]

4+8 != 16

Esercizio 3

```
typedef struct Tdato{
  int dato;   char *c;
}Tdato;
int main(){
 Tdato *v2;
 v2 = new Tdato[10];
 for(int i=0; i<10; i++){
  v2[i].dato = 0;
  v2[i].c = new char[5];
  (*( v2+i )).c = "abcd"; //deprecato!!!
  //*( v2+i )  →  v2[i]
  //( *( v2+i ) ).dato  →  v2[i].dato
 }
 for(int i=0; i<10; i++) {
   delete[] v2[i].c;
 }
 delete[] v2;
}
```

DEALLOCAZIONE

Esercizio 4

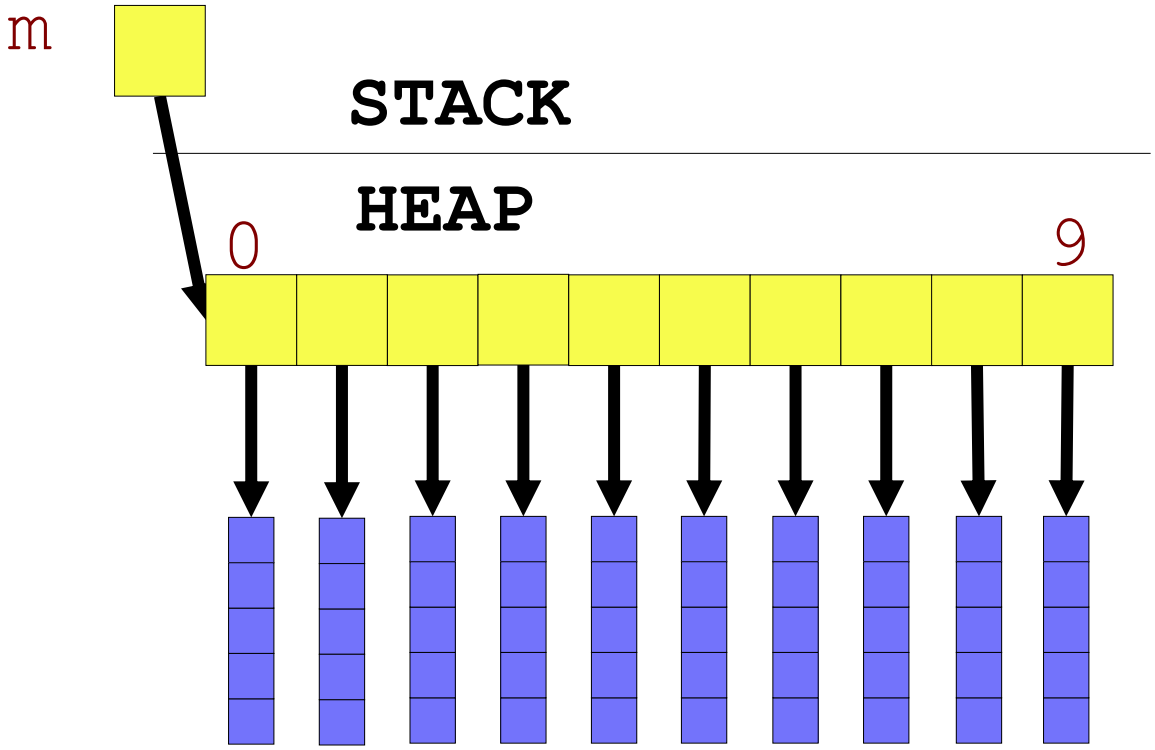
Scrivere un programma che:

- Alloca una matrice di interi 10x5
  - Allocazione dinamica
- Inizializza la matrice con valori casuali
- Stampa a video il contentuto della matrice

- `int **m;`

```
int **m;
m = new int*[10];
for(int i=0; i<10 ; i++){
    *(m+i) = new int[5];
```

Allocazione memoria (Esercizio 4)



Le dimensioni sono funzione anche dell'architettura del sistema

Esercizio 4

```
int **m;
m = new int*[10];
for(int i=0; i<10 ; i++){
    *(m+i) = new int[5];
    //*( m+i )  → m[i]
    //*( *( m+i ) +j )  → m[i][j]
}
//...
```

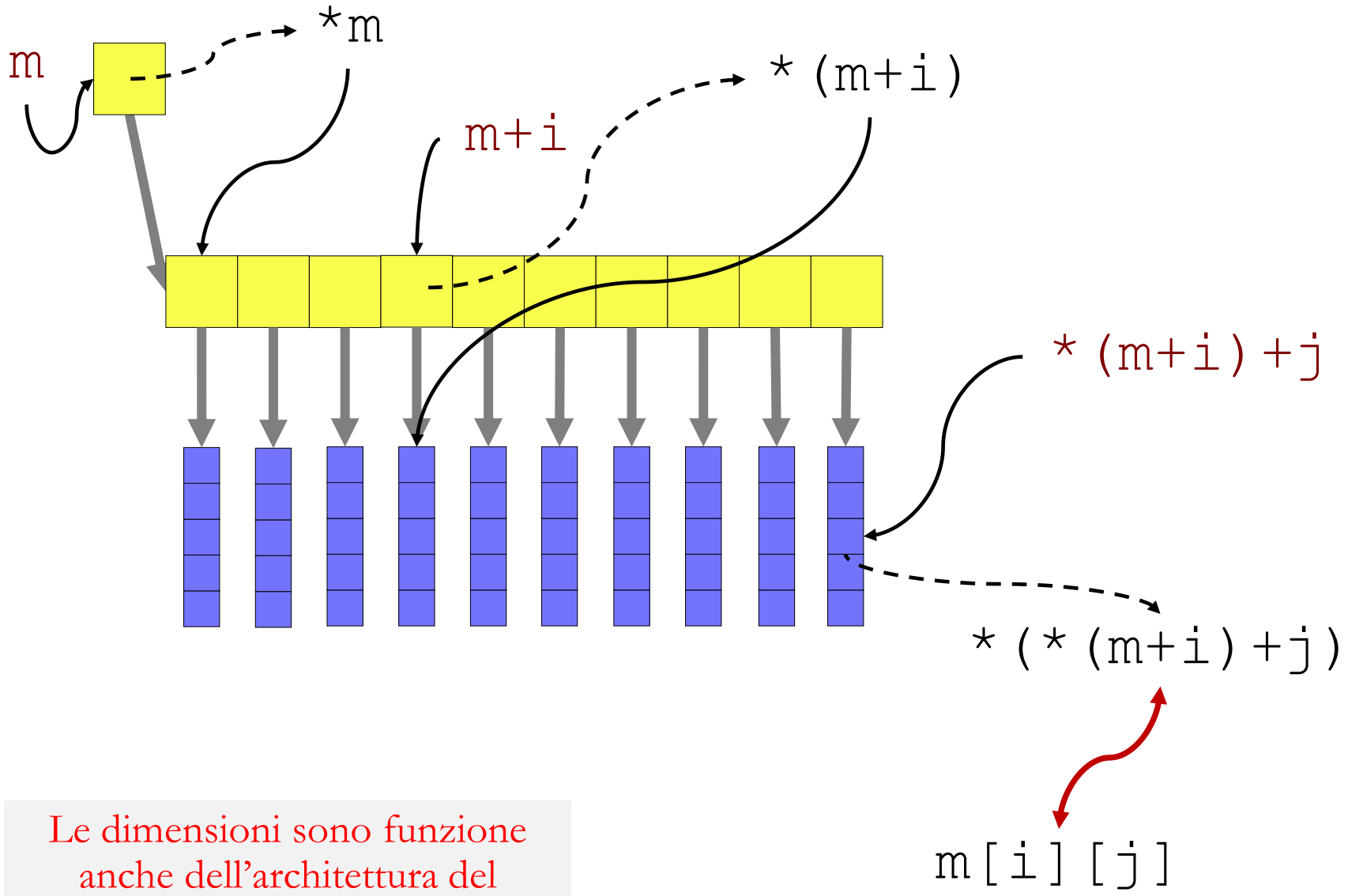
Esercizio 4 – con passaggio per indirizzo

```
int main(){
    …
    int** m;
    init(&m);
}
void init(int** * m) {
    *m = new int*[10]; //deferenziazione
    for(int i=0; i<10 ; i++){
        (*m)[i]= new int[5];
    } …
}
```

Per analogia con una variabile che non è un puntatore:

```
int main() {
    …
    int a;
    init(&a);
}
void main(int *a) { … }
```

Allocazione memoria (Esercizio 4)



Le dimensioni sono funzione anche dell'architettura del sistema

Esercizio 4

```
//disallocazione

for(int i=0; i<MAXR ; i++){
    delete[] *(m+i);
    // delete[] m[i];
    // delete *(m+i);
    // delete m[i];
}
delete[] m;
```

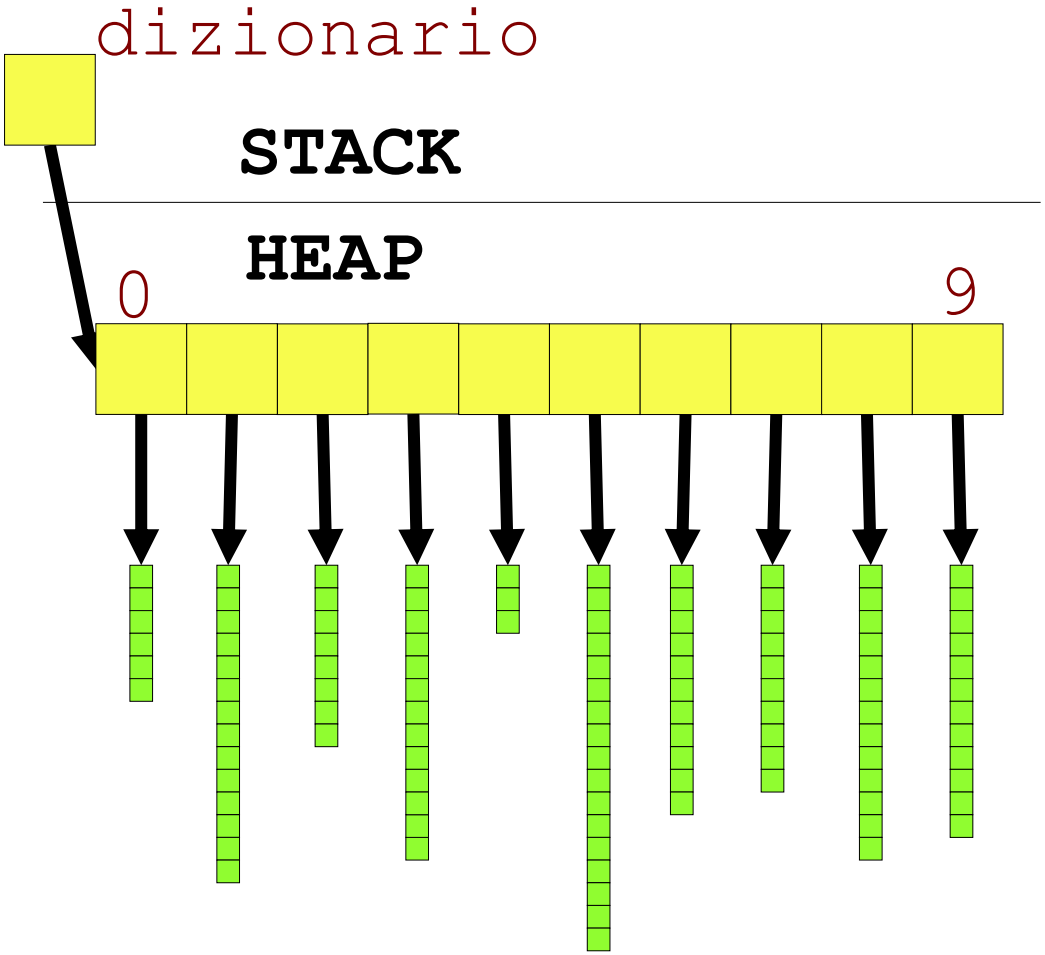
Esercizio 5

Scrivere un programma che:

- Alloca un array dizionario di puntatori a carattere
  - Dimensione 10
  - Allocazione dinamica
- Per ogni elemento i dell'array dizionario
  - Chiede all'utente di inserire una stringa
  - Memorizza la stringa nella posizione dizionario[i]
- Stampa a video le stringhe lette

- `char **dizionario;`

Allocazione memoria (Esercizio 5)



Le dimensioni sono funzione anche dell'architettura del sistema

Esercizio 5

```
char **dizionario;
dizionario = new char*[10];

for(int i =0 ; i<10 ; i++){
    //allocazione fissa di 10 caratteri
    dizionario [i] = new char[101];
    fflush(stdin);
    scanf("%s", dizionario[i]);
}
```

Esercizio 5

```
char **dizionario;
dizionario = new char*[10];
char tmp[101];

for(int i =0 ; i<10 ; i++){
  //allocazione esatta in base alla
  //dimensione del dato in input
  scanf("%s", tmp);
  dizionario[i] = new char[ strlen(tmp)+1 ];
  strcpy(*(dizionario+i), tmp);
  //strcpy(dizionario[i], tmp);
}
```

Esercizio 5

```
//de-allocazione

for(int i =0 ; i<10 ; i++){
    delete[] dizionario[i];
}
delete[] dizionario;
```

Esercizio 6

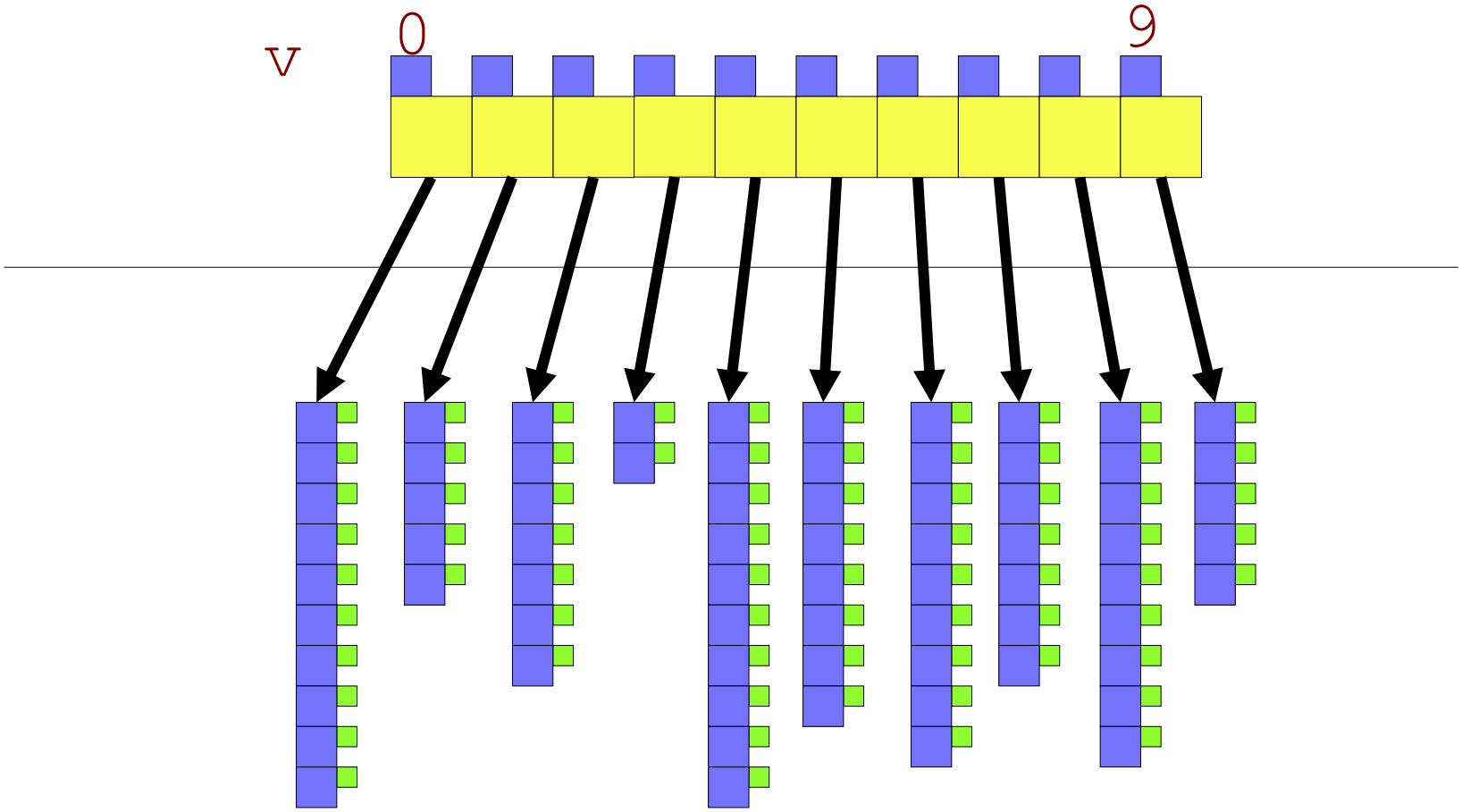
Scrivere un programma che:

- Considerate le seguenti strutture

```
Tdato: int k; char c;
Tnodo: int n; Tdato *p;
```

- Definire un array v di elementi di tipo Tnodo di dimensione 10 (usare costante N)
  - Allocazione **statica**
- Per ogni elemento di v
  - Inizializzare il campo n in modo casuale (2 - 8)
  - Allocare il vettore p con dimensione pari al valore del campo n (Allocazione **dinamica**)
- Inizializzare i campi k e c con valori casuali (0-99, a-z)
- Stampare il contenuto della struttura dati

Allocazione memoria (Esercizio 6)



Le dimensioni sono funzione anche dell'architettura del sistema

Esercizio 7

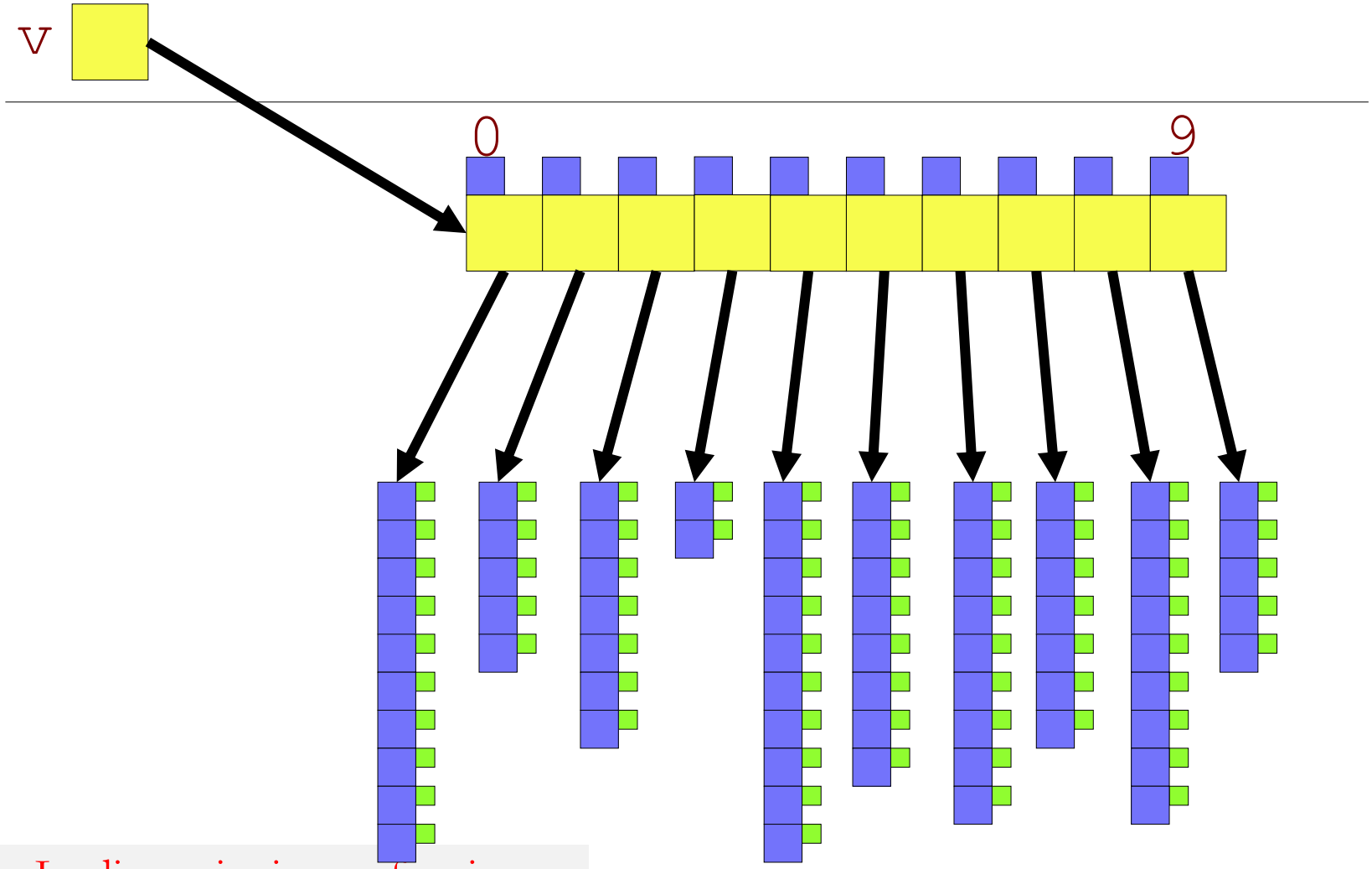
Scrivere un programma che:

- Considerate le seguenti strutture

```
Tdato: int k; char c;
Tnodo: int n; Tdato *p;
```

- Definire un array v di elementi di tipo Tnodo di dimensione 10 (usare costante N)
  - Allocazione **dinamica**
- Per ogni elemento di v
  - Inizializzare il campo n in modo casuale (2 - 8)
  - Allocare il vettore p con dimensione pari al valore del campo n (Allocazione **dinamica**)
- Inizializzare i campi k e c con valori casuali (0-99, a-z)
- Stampare il contenuto della struttura dati

Allocazione memoria (Esercizio 7)



Le dimensioni sono funzione anche dell'architettura del sistema

Funzione che restituisce un array

- Una funzione **non può restituire direttamente** un array ma **può restituire un puntatore al primo elemento dell'array**
- Un errore tipico che si commette in tale situazione è la restituzione di un puntatore ad una variabile locale che viene rimossa dalla memoria al terminare della funzione:

```
int *times_stack(int a[10], int k) {
   int b[10];
   for (int i=0; i<10; i++)
      b[i]=a[i]*k;
   return b; // errore b è locale!
}
```

Funzione che restituisce un array

```
int *times_heap(int a[10], int k) {
    int * b = new int [10];
    for (int i=0; i<10; i++)
        b[i]=a[i]*k;
    // responsabilità del chiamante
    // deallocare memoria puntata da b
    return b;
}
```

Esercizi aggiuntivi

Esercizio A1 – 1/2 stringhe

Scrivere un programma che:

- Chiede all'utente di scegliere un'operazione
  - 1: calcolare lunghezza di una stringa
  - 2: comparare due stringhe
  - 3: copiare una stringa
  - 0: fine
- Se 1
  - Chiedere l'inserimento della stringa, calcolare la lunghezza e stampare a video la lunghezza
- Se 2
  - Chiedere l'inserimento di due stringhe e stampare a video informazione se le stringhe sono uguali oppure quella minore

Esercizio A1 – 2/2 stringhe

- Se 3
  - Chiedere all'utente una stringa, copiarla in una nuova stringa e stamparla a video
- Se 0
  - Termina il programma
- Prototipi funzioni
  - `int cstrlen(char *str);`
    - return: lunghezza stringa
  - `int cstrcmp(char *str1, char *str2);`
    - return: 0 se uguali, <0 se str1<str2, >0 se str1>str2
  - `void cstrcpy(char *dst, char *src);`
  - `void strprint(char* str) ;`
    - utilizza printf con descrittore formato %c

Esercizio A2 – 1/3

- Scrivere il seguente programma

```
#include <cstdlib>
#include <iostream>
#define LEN 10
using namespace std;

//calcolo la lunghezza della sequenza di char in memoria
int mystrlen(char* s){
   int len=0;
   while(s[len]){len++;}
   return len;
}
int main(int argc, char *argv[]){
    char* s1="Stringa1";
    char s2[100];
    char s3[100];
    char s4[]="Stringa4";
    char* s5;
    char* s6;
    int i;
    //…
```

Esercizio A2 – 2/3

```
int main(int argc, char *argv[]){
   //…
    for(i=0 ; i<LEN ; i++) {
       s2[i]='a'+i;
    }
    for(i=0 ; i<LEN ; i++) {
       s3[i]='a'+i;
    }
    s3[i]='\0';
    s5 = new char[100];
    for(i=0 ; i<LEN ; i++) {
       s5[i]='a'+i;
    }
    s6 = new char[100];
    for(i=0 ; i<LEN ; i++) {
       s6[i]='a'+i;
    }
    s6[i]='\0';
}
```

Esercizio A2 – 3/3

```
    printf("\nStringhe\n");
    printf("s1: %s\n", s1);
    printf("s2: %s\n", s2);
    printf("s3: %s\n", s3);
    printf("s4: %s\n", s4);
    printf("s5: %s\n", s5);
    printf("s6: %s\n", s6);

    printf("\nLunghezze\n");
    printf("dim s1: %d\n", mystrlen(s1));
    printf("dim s2: %d\n", mystrlen(s2));
    printf("dim s3: %d\n", mystrlen(s3));
    printf("dim s4: %d\n", mystrlen(s4));
    printf("dim s5: %d\n", mystrlen(s5));
    printf("dim s6: %d\n", mystrlen(s6));
    delete s5;
    delete s6;
    system("PAUSE");
    return 0;
}
```

Esercizio A3

Scrivere un programma che:

1. apre un file binario (`FILE1`) in scrittura
2. chiede la dimensione di un array di interi `array1` (minore di MAX e maggiore di 1)
3. alloca **dinamicamente** `array1`
4. popola l'array `array1` con numeri casuali. Il popolamento viene fatto con una funzione che modifica in contenuto del puntatore a tale struttura
   `void popola(int *a, int dim);`
5. scrive nel file binario (`FILE1`) gli elementi di `array1`

Esercizio A4 – 1/3

- Database di Aziende
- Struttura dati

```
typedef struct {
  char* nome;
  char codice[6];
}tAzienda;

typedef struct {
  tAzienda v[100];
  int n_azienda;
}tDB;
```

Esercizio A4 – 2/3

- Il programma deve chiedere all'utente il nome e il codice di 5 aziende.
- Salvare i dati nelle strutture in modo opportuno
- Chiedere all'utente il nome del file dove salvare i dati
- Salvare i dati nel file indicato in formato **testo**
- Liberare la memoria utilizzata nell'allocazione dinamica

- Utilizzare funzioni
- Suddividere il programma in file.h e file.c (+ main.c)

Esercizio A4 – 3/3

```
   /* nessuna azienda in DB */
void inizializza_database(tDB* pdb);

   /* inizializza azienda (primo param) con i succesivi parametri forniti */
void inizializza_azienda(tAzienda* pa, char* nome, char* codice);

  /* aggiungere azienda definita da nome e codice nel database */
void aggiungi_azienda(tDB* p, char* nome, char* codice);

  /* free di allocazioni dinamiche */
void cancella_database(tDB* p);

  /* stampa dei dati delle aziende presenti nel database */
void stampa(tDB elem);

  /* salvare il database elem su file (identificato da nome nome_file) -
   formato TESTO */
void salva_su_file(tDB elem, char* nome_file);
```

Esercizio A4 – funzioni utili

- Libreria funzioni gestione stringhe
  - **#include <cstring>** // c++
  - oppure
  - **#include <string.h>** // c

- **int strlen(char* s);**
  - Restituisce la lunghezza stringa s

- **void strcpy(char* dst, char* src);**
  - Copia la stringa sorgente (src) nella stringa destinazione (dst)

- **FILE* fopen(char* nome_file, char* mode);**
  - Apre file in modalità mode. Return NULL se errore

- **int fprintf(FILE* f, ...);**
  - Stampa su file identificato da stream f

Esercizio A4 – Note

- Era possibile salvare la struttura tDB in formato binario?
  Ad esempio:

```
fwrite(&database, 1, sizeof(tDB), pfout);
```

Esercizio A5 – estensione Es A3

Estendere l'esercizio 2 in modo:

5. creare un array di interi `array2` di **MAX** elementi (MAX =50)
6. inizializza gli elementi di `array2` come `array2[indice] = 10 * indice`
7. crea un file binario (`FILE2`) il quale contiene 50 elementi di `array2`
8. apre i file appena creati (`FILE1 FILE2`) in lettura
9. chiede all'utente un numero (<50) `I`
10. legge l'i-esimo elemento da entrambi i file (dove `i` è il valore inserito da tastiera dall'utente)

- continuare a chiedere all'utente quale elemento `i` si deve leggere
- un valore di `i` < 0 significa terminare il programma
- usare la funzione **`fseek`**

Esercizio A5 – fseek

fseek(FILE* pfile, long int offset, int origin);

Origin:
SEEK_SET Inizio file
SEEK_CUR Posizione corrente del puntatore
SEEK_END Fine del file

```
int k;
FILE *f;
// file binario in cui sono scritti 100 interi
…
//riportare puntatore file all'inizio
fseek(f, 0, SEEK_SET);
…
//portare puntatore file alla posizione off
fseek(f, sizeof(int)*off, SEEK_SET);
// leggere elemento in posizione off
fread(&k, sizeof(int), 1, f);
```