

#### Laurea triennale in Informatica

modulo (CFU 6) di

## Programmazione II e Lab.

# prof. Mariarosaria Rizzardi

Centro Direzionale di Napoli – Isola C4 stanza: n. 423 – IV piano Lato Nord

tel.: 081 547 6545

email: mariarosaria.rizzardi@uniparthenope.it



### VLA: variable length array

A partire dall'ISO C99 sono stati introdotti gli "array a lunghezza variabile" (Variable Length Array - VLA).

Questi sono array il cui size non è noto al tempo della compilazione, bensì durante l'esecuzione.

Il compilatore GNU gcc alloca queste variabili nella memoria stack.

L'accesso a tale memoria risulta più efficiente di quello alla memoria heap, che richiede l'uso dei puntatori.

#### Esempio 1a

dal C99 in poi

```
#include <stdlib.h>
                                       #include <stdlib.h>
                                       #include <stdio.h>
#include <stdio.h>
#define N 8
                                       int main()
int main()
                                           unsigned int N;
    int a[N];
                                            printf("Enter N>0: ");
                                            scanf("%u", &N);
    for (int k=0; k<N; k++)
        a[k] = rand()%10;
                                           int a[N];
                                           for (int k=0; k<N; k++)
    for (int k=0; k<N; k++)
                                                a[k] = rand()%10;
        printf("%d\n",a[k]);
    return 0;
                                           for (int k=0; k<N; k++)
                                                printf("%d\n",a[k]);
                                           return 0;
```

Il size N dell'array è noto al tempo della compilazione

Il size N dell'array è noto solo al tempo dell'esecuzione

#### Esempio 1a C++

```
#include <iostream>
#include <random>
using namespace std;
 int main()
    std::default_random_engine generator;
    std::uniform_int_distribution<int> distribution(0,9);
    unsigned int N;
    cout << "Enter N>0: "; cin >> N; ←
    int a[N];
    for (int k=0; k<N; k++)
        a[k] = distribution(generator); // a[k]=rand()%10;
    for (int k=0; k<N; k++)
        cout << a[k] << endl;</pre>
                                  Il size dell'array è noto solo
     return 0;
                                  al tempo dell'esecuzione
```

#### Esempio 1b

dal C99 in poi

```
#include <stdlib.h>
#include <stdio.h>
#define M 5
#define N 8
int main()
  int A[M][N];
  for (int i=0; i<M; i++)
     for (int j=0; j<N; j++)
        A[i][j] = rand()%10;
  for (int i=0; i<M; i++) {
    for (int j=0; j<N; j++)
      printf(" %d",A[i][j]);
    puts("");
    return 0;
```

#include <stdlib.h> #include <stdio.h> int main() { unsigned int M, N; printf("Enter M>0: "); scanf("%u", &M); printf("Enter N>0: "); scanf("%u", &N); int A[M][N]; for (int i=0; i<M; i++) for (int j=0; j<N; j++) A[i][j] = rand()%10;for (int i=0; i<M; i++) { for (int j=0; j<N; j++) printf(" %d", A[i][j]); puts(""); return 0;

Il size M×N dell'array è noto al tempo della compilazione

Il size M×N dell'array è noto solo al tempo dell'esecuzione

#### Esempio 2

```
#include <stdlib.h>
                                             int *createArray(unsigned int N)
#include <stdio.h>
                                                   int a[N];
int *createArray(unsigned int N);
                                                   for (int k=0; k<N; k++)
                                                        a[k] = rand()%10;
int main()
                                                   return a;
     unsigned int N;
     printf("Enter N>0:
     scanf("%u", &N);
                                        La compilazione segnala: "warning: function
                                        returns address of local variable", ma
     int *b:
                                        l'esecuzione ... va fuori della memoria lecita
     b = createArray(N);
                                                              vla2.exe
     for (int k=0; k<N; k++)
                                           vla2.exe ha smesso di funzionare
          printf("%d\n",b[k]);
                                           Windows: si è verificato un problema che impedisce il funzionamento corretto
                                           del programma. Se è disponibile una soluzione, verrà chiuso il programma e
     return 0;
                                           inviata una notifica automatica.
                                             segmentation fault
                                                                           Chiudi programmi
```

#### Dove il programma va fuori memoria? E perché?

#### Esercizio

Come eliminare l'errore di "segmentation fault" nel precedente programma?

```
#include <stdlib.h>
#include <stdio.h>
int *createArray(unsigned int N);
                              int *createArray(unsigned int N)
int main()
    unsigned int N;
                                  int a[N];
    printf("Enter N>0: ");
                                  for (int k=0; k<N; k++)
    scanf("%u", &N);
                                       a[k] = rand()%10;
    int *b;
                                  return a;
    b = createArray(N);
    for (int k=0; k<N; k++)
        printf("%d\n",b[k]);
    return 0;
```

#### Esempio 3

```
#include <stdlib.h>
#include <stdio.h>
int *createArray(unsigned int N);
int main()
    unsigned int N;
    printf("Enter N>0: ");
    scanf("%u", &N);
    int *b;
    b = createArray(N);
    for (int k=0; k<N; k++)
        printf("%d\n",b[k]);
    return 0;
```

```
int *createArray(unsigned int N)
{
   int *a=(int*)malloc(N*sizeof(int));
   if (a == NULL) exit(-1);

   for (int k=0; k<N; k++)
        a[k] = rand()%10;
   return a;
}</pre>
```

## Ora funziona!

Perché?