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

Il compilatore GNU gcc e i Variable Length Array [VLA]

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


```c
#include <stdlib.h>
#include <stdio.h>
#define N 8

int main()
{
    int a[N];
    for (int k=0; k<N; k++)
        a[k] = rand()%10;

    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

dal C99 in poi

```c
#include <stdlib.h>
#include <stdio.h>

int main()
{
    unsigned int N;
    printf("Enter N>0: ");
    scanf("%u", &N);

    int a[N];        VLA
    for (int k=0; k<N; k++)
        a[k] = rand()%10;

    for (int k=0; k<N; k++)
        printf("%d\n",a[k]);

    return 0;
}
```

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];        VLA
    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;

     return 0;
 }
```

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

Esempio 1b

```c
#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;
}
```

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

dal C99 in poi

```c
#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];   VLA
  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 solo al tempo dell'esecuzione

Esempio 2

```c
#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;
}
```

```c
int *createArray(unsigned int N)
{
    int a[N];          VLA
    for (int k=0; k<N; k++)
        a[k] = rand()%10;
    return a;
}
```

La compilazione segnala: "warning: function returns address of local variable", ma l'esecuzione ... va *fuori della memoria lecita*

Dove il programma va fuori memoria? E perché?

Esercizio

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

```c
#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;
}
```

```c
int *createArray(unsigned int N)
{
    int a[N];
    for (int k=0; k<N; k++)
        a[k] = rand()%10;
    return a;
}
```

Esempio 3

```c
#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;
}
```

```c
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;
}
```

Ora funziona!

Perché?