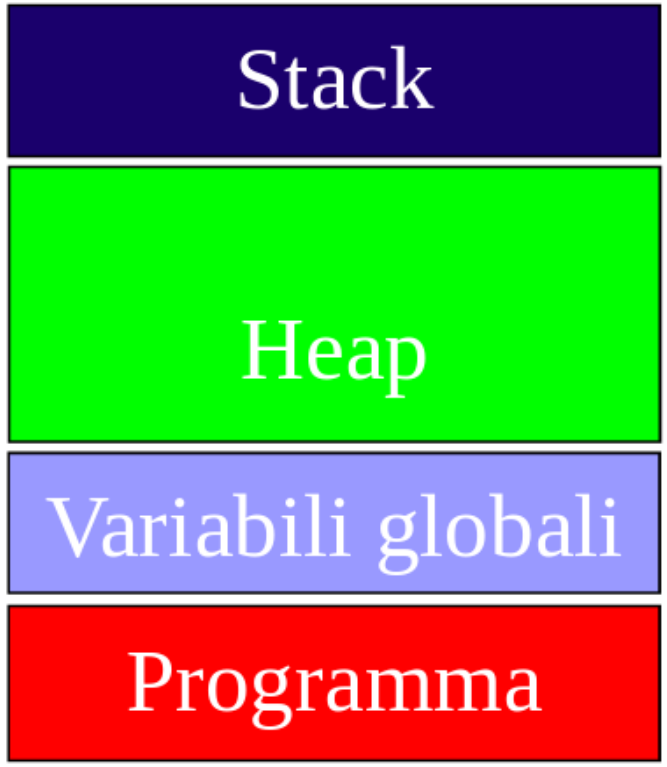
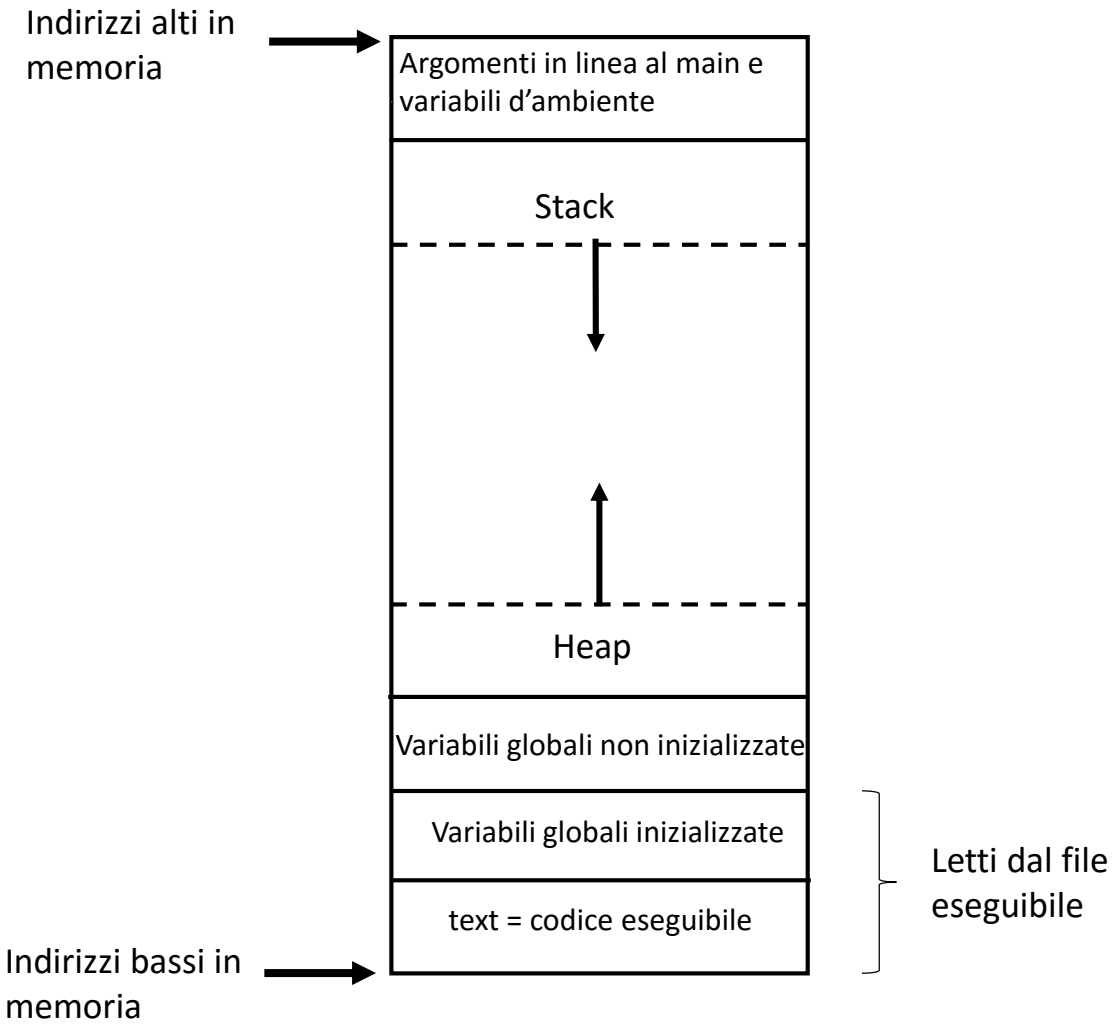
Note sulla struttura della memoria

EM

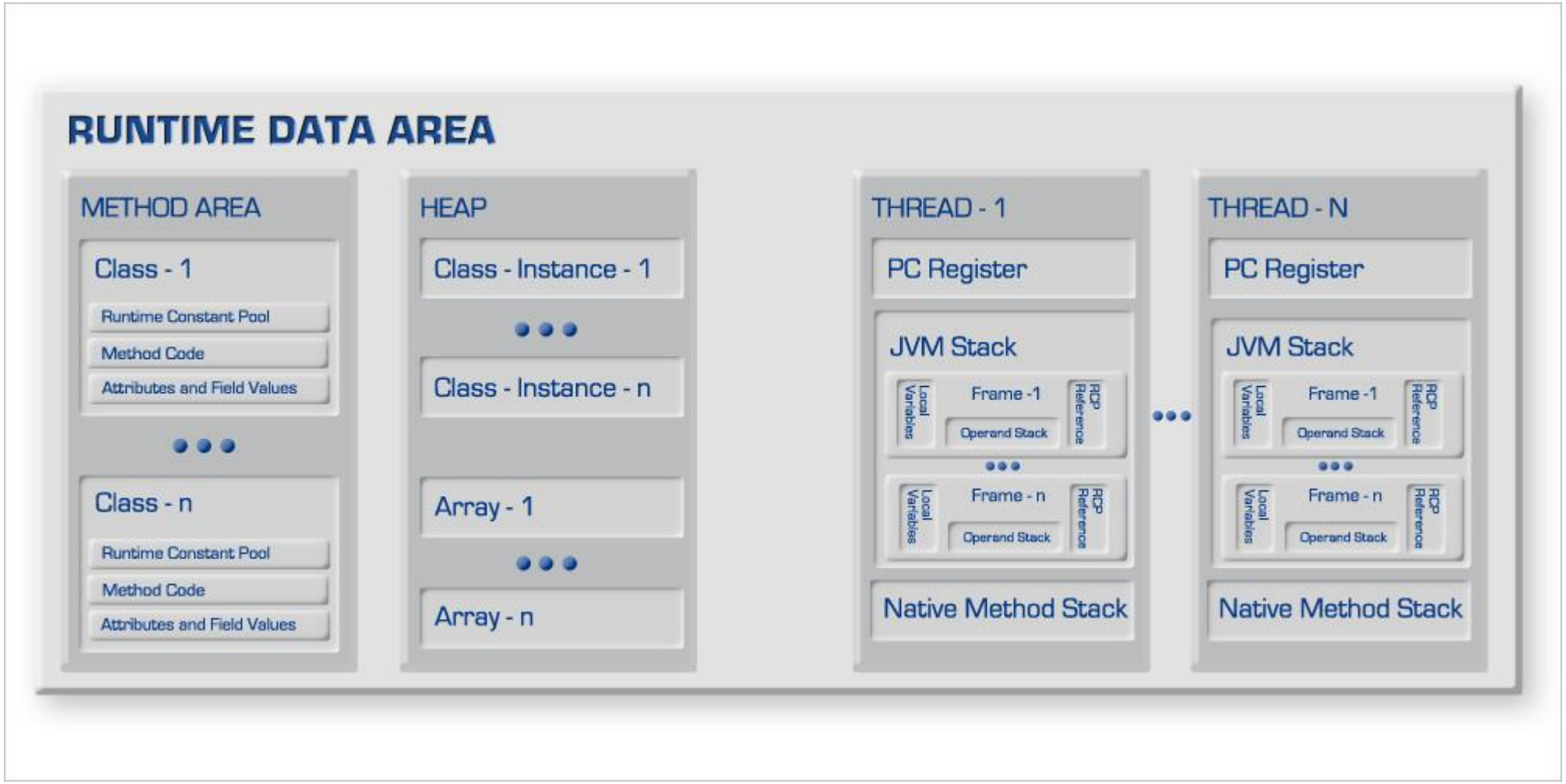
Struttura di un eseguibile in linguaggio C



Più in dettaglio



Struttura della memoria di un eseguibile Java



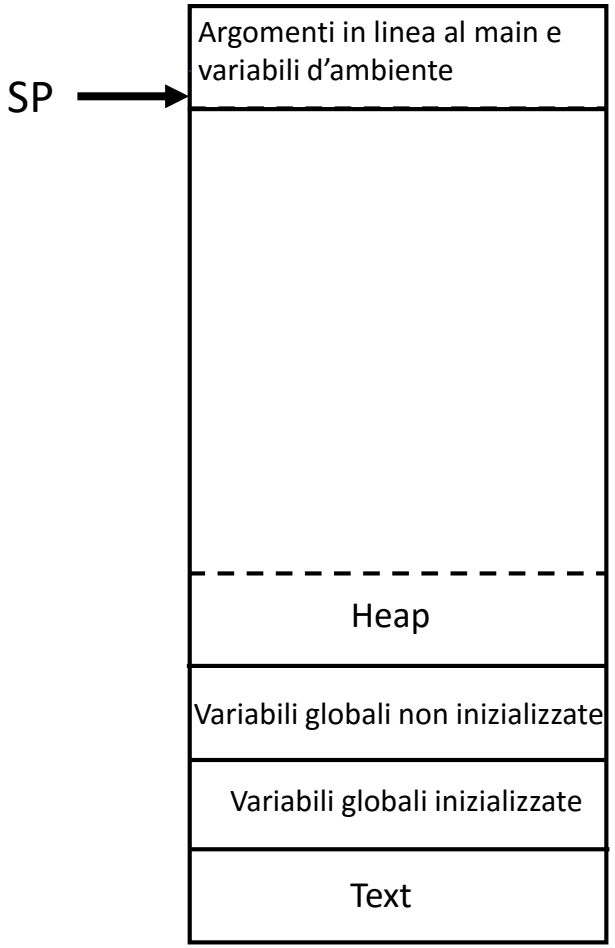
Semplice programma C

```
/* file simple1.c*/
/* include macro e variabili */
#include <stdio.h>
#include <stdlib.h>

void func1(int); /* definizione funzioni */

int main(){
    int i;     /* variabile locale */
    int count; /* variabile globale */
    for(i=0; i<10; i++) {
        count=i*2;
        func1(count);
    }
    return 0;
}

void func1(int c){ /* c variabile formale */
    printf("count=%d\n",c);
}
```

In esecuzione

PC →

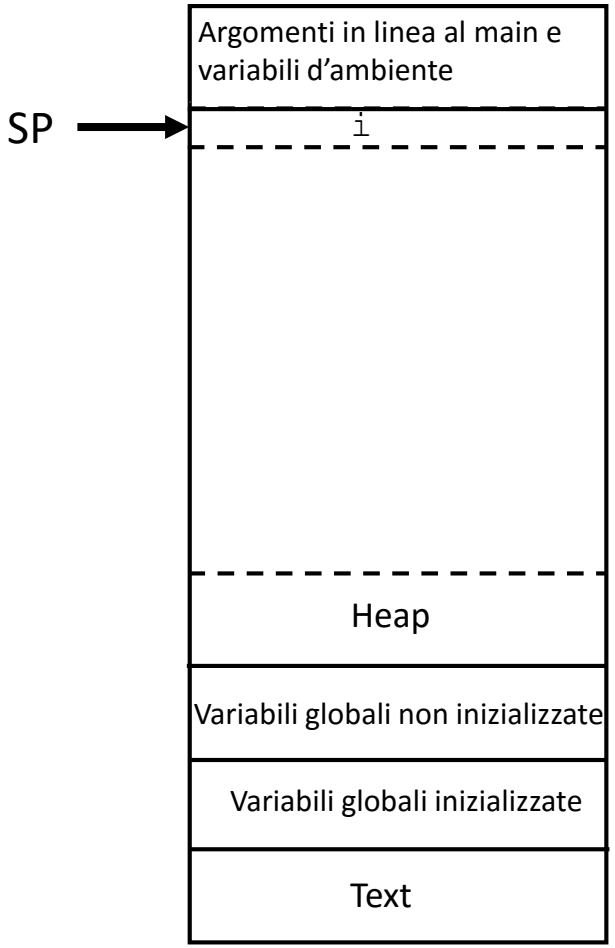
```
/* file simple1.c*/
/* include macro e variabili */
#include <stdio.h>
#include <stdlib.h>

void func1(int); /* definizione funzioni */

int main(){
    int i;     /* variabile locale */
    int count; /* variabile globale */
    for(i=0; i<10; i++) {
        count=i*2;
        func1(count);
    }
    return 0;
}

void func1(int c){ /* c variabile formale */
    printf("count=%d\n",c);
}
```



In esecuzione

PC →

```
/* file simple1.c*/
/* include macro e variabili */
#include <stdio.h>
#include <stdlib.h>

void func1(int); /* definizione funzioni */

int main(){
    int i;     /* variabile locale */
    int count; /* variabile globale */
    for(i=0; i<10; i++) {
         count=i*2;
         func1(count);
    }
    return 0;
}

void func1(int c){ /* c variabile formale */
    printf("count=%d\n",c);
}
```



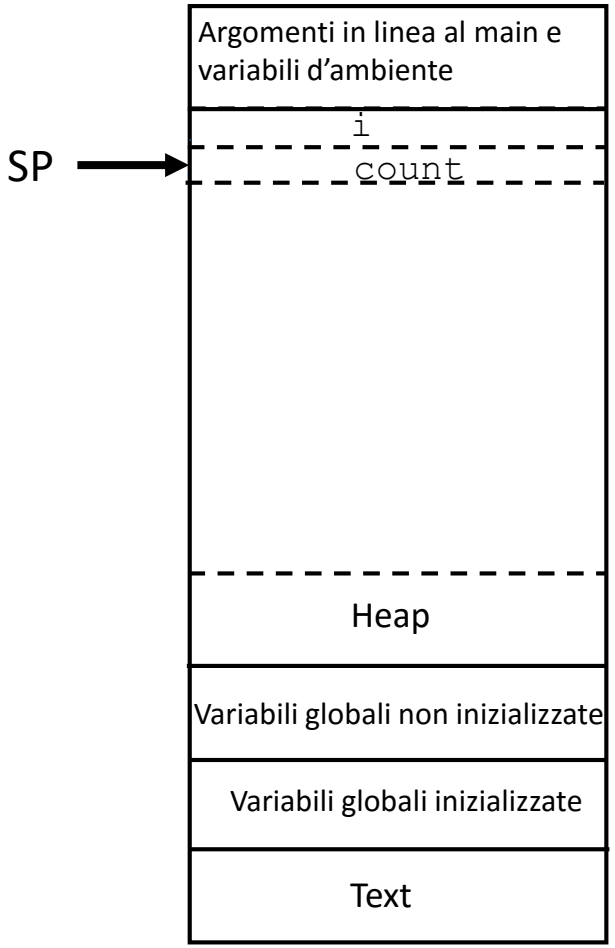
Allocazione variabile locale i:

SP- -
Indirizzo di 'i' = SP

In esecuzione

PC →

```
/* file simple1.c*/
/* include macro e variabili */
#include <stdio.h>
#include <stdlib.h>

void func1(int); /* definizione funzioni */

int main(){
    int i;     /* variabile locale */
    int count; /* variabile globale */
    for(i=0; i<10; i++) {
         count=i*2;
         func1(count);
    }
    return 0;
}

void func1(int c){ /* c variabile formale */
    printf("count=%d\n",c);
}
```



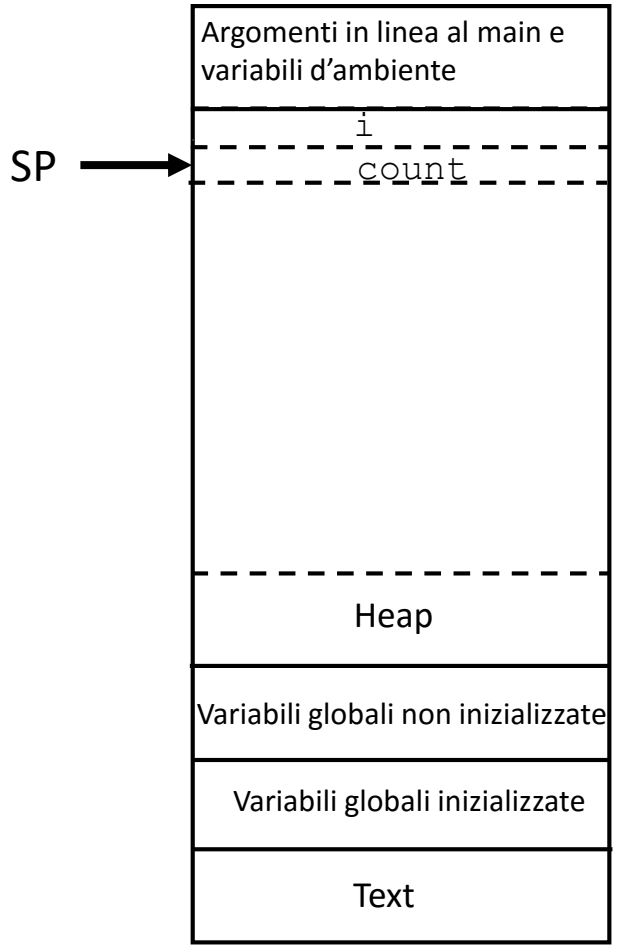
Allocazione variabile locale:

SP- -
Indirizzo di 'count' = SP

In esecuzione

PC →

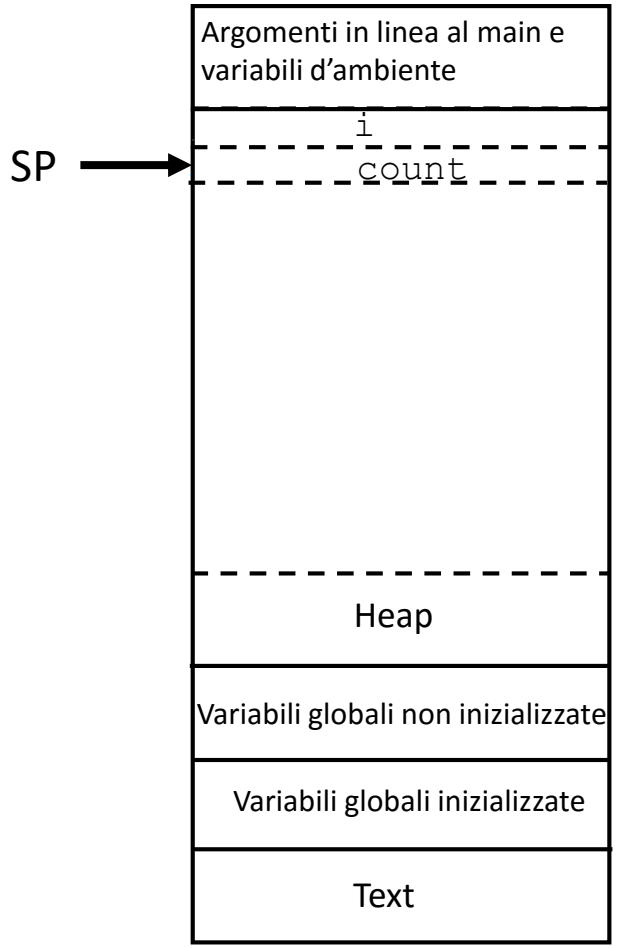
```
/* file simple1.c*/
/* include macro e variabili */
#include <stdio.h>
#include <stdlib.h>

void func1(int); /* definizione funzioni */

int main(){
    int i;     /* variabile locale */
    int count; /* variabile globale */
    for(i=0; i<10; i++) {
        count=i*2;
        func1(count);
    }
    return 0;
}

void func1(int c){ /* c variabile formale */
    printf("count=%d\n",c);
}
```



In esecuzione

PC →

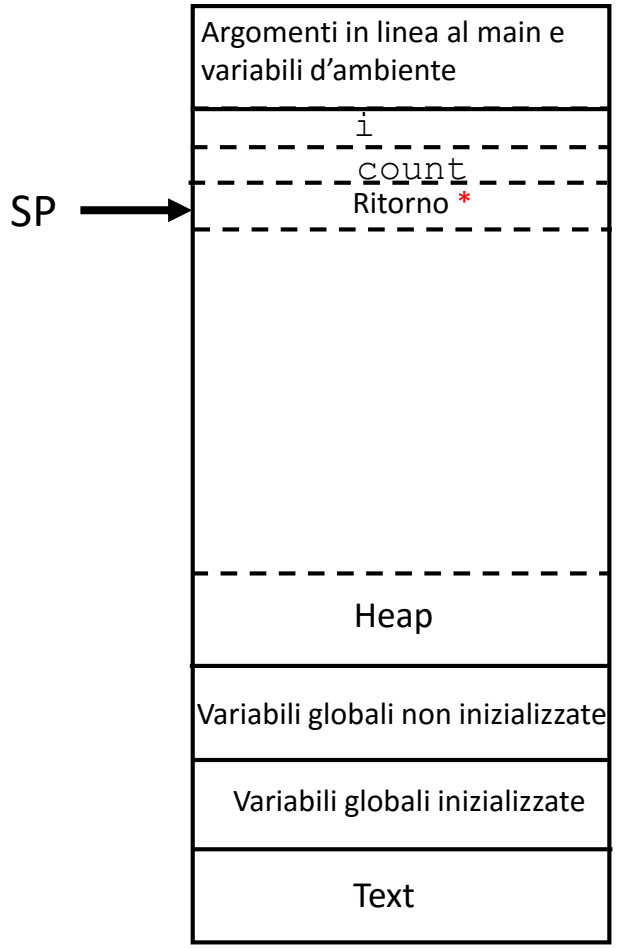
```
/* file simple1.c*/
/* include macro e variabili */
#include <stdio.h>
#include <stdlib.h>

void func1(int); /* definizione funzioni */

int main(){
    int i;     /* variabile locale */
    int count; /* variabile globale */
    for(i=0; i<10; i++) {
        count=i*2;
        func1(count);
    }
    return 0;
}

void func1(int c){ /* c variabile formale */
    printf("count=%d\n",c);
}
```



In esecuzione

PC →

```
/* file simple1.c*/
/* include macro e variabili */
#include <stdio.h>
#include <stdlib.h>

void func1(int); /* definizione funzioni */

int main(){
    int i;     /* variabile locale */
    int count; /* variabile globale */
    for(i=0; i<10; i++) {
        count=i*2;
        func1(count);
    }
    return 0;
}

void func1(int c){ /* c variabile formale */
    printf("count=%d\n",c);
}
```



Allocazione indirizzo di ritorno:

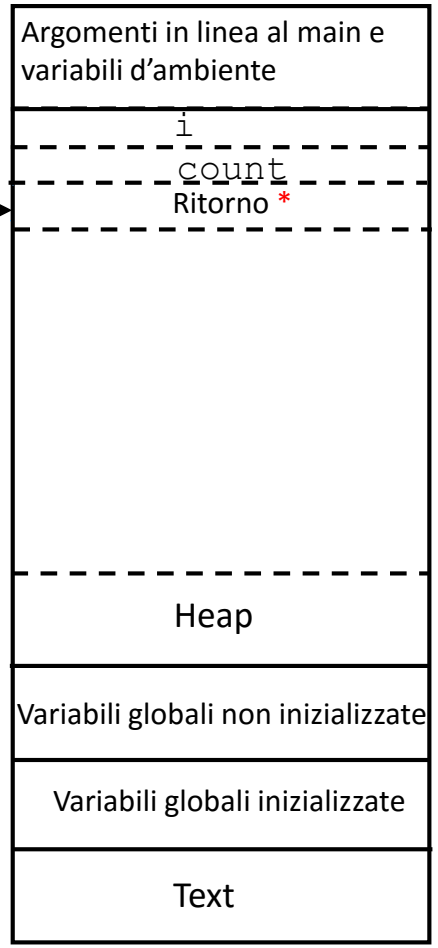
SP- -
(SP) = *
PC=indirizzo di func1()

In esecuzione

```
/* file simple1.c*/
/* include macro e variabili */
#include <stdio.h>
#include <stdlib.h>

void func1(int); /* definizione funzioni */

int main(){
    int i;     /* variabile locale */
    int count; /* variabile globale */
    for(i=0; i<10; i++) {
         count=i*2;
         func1(count);
    }
    return 0;
}

void func1(int c){ /* c variabile formale */
    printf("count=%d\n",c);
}
```

PC →

SP →

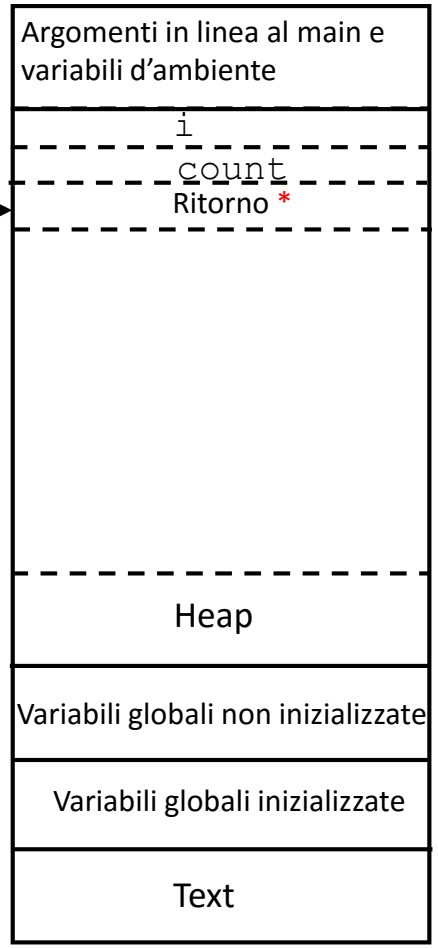


In esecuzione

```
/* file simple1.c*/
/* include macro e variabili */
#include <stdio.h>
#include <stdlib.h>

void func1(int); /* definizione funzioni */

int main(){
    int i;     /* variabile locale */
    int count; /* variabile globale */
    for(i=0; i<10; i++) {
         count=i*2;
         func1(count);
   *}
    return 0;
}

void func1(int c){ /* c variabile formale */
    printf("count=%d\n",c);
}
```

PC →

SP →



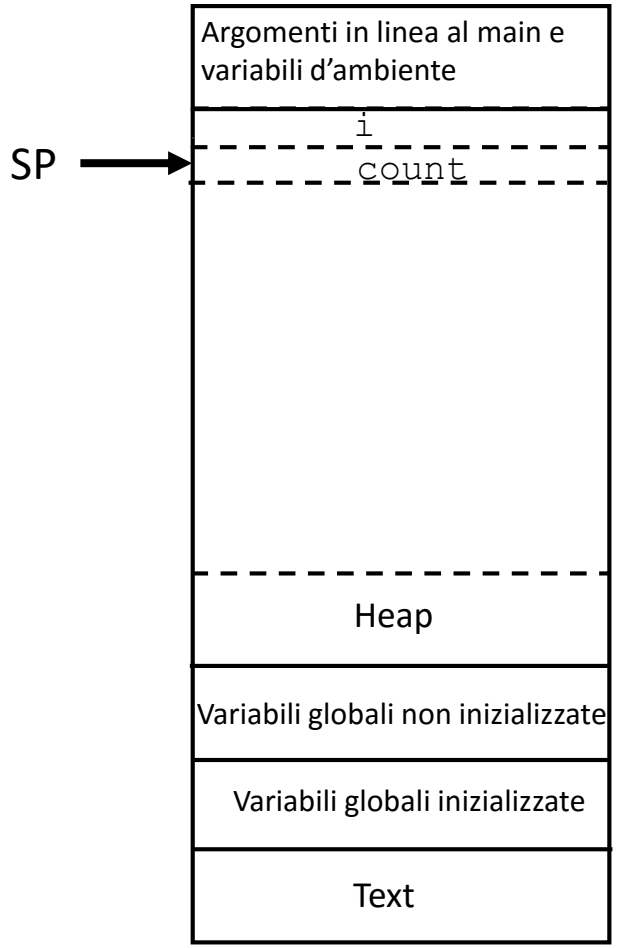
Alla terminazione di func1()

PC=(SP)
SP++

In esecuzione

PC →

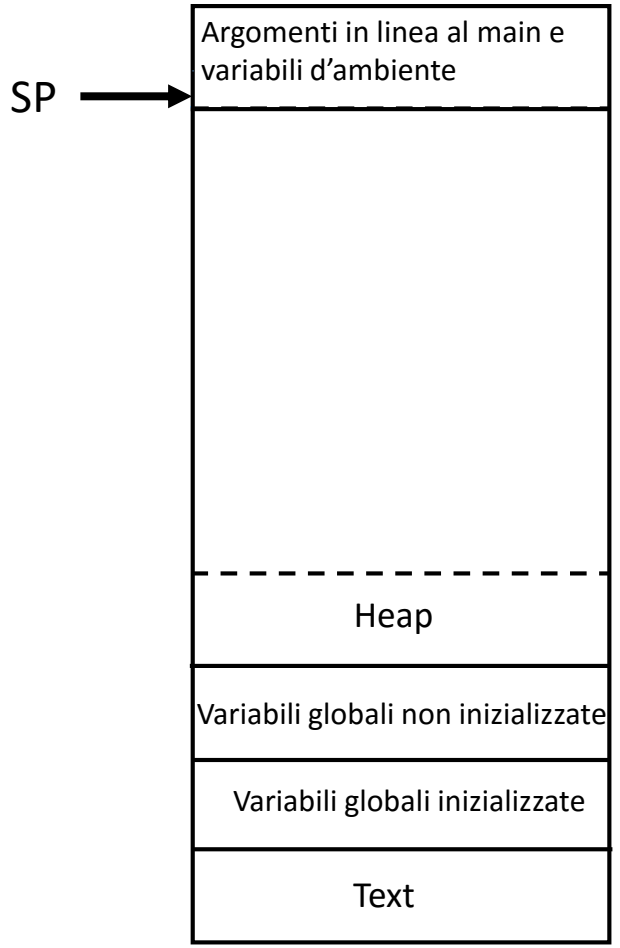
```
/* file simple1.c*/
/* include macro e variabili */
#include <stdio.h>
#include <stdlib.h>

void func1(int); /* definizione funzioni */

int main(){
    int i;     /* variabile locale */
    int count; /* variabile globale */
    for(i=0; i<10; i++) {
        count=i*2;
        func1(count);
    }
    return 0;
}

void func1(int c){ /* c variabile formale */
    printf("count=%d\n",c);
}
```



Dopo 10 chiamate di `func1()`

```
/* file simple1.c*/
/* include macro e variabili */
#include <stdio.h>
#include <stdlib.h>

void func1(int); /* definizione funzioni */

int main(){
    int i;     /* variabile locale */
    int count; /* variabile globale */
    for(i=0; i<10; i++) {
        count=i*2;
        func1(count);
  * }
    return 0;
}

void func1(int c){ /* c variabile formale */
    printf("count=%d\n",c);
}
```

PC →



Alla terminazione del main()

SP++
SP++

LO STACK POINTER TORNA AL PUNTO DI PARTENZA