Esercitazione di Reti degli elaboratori

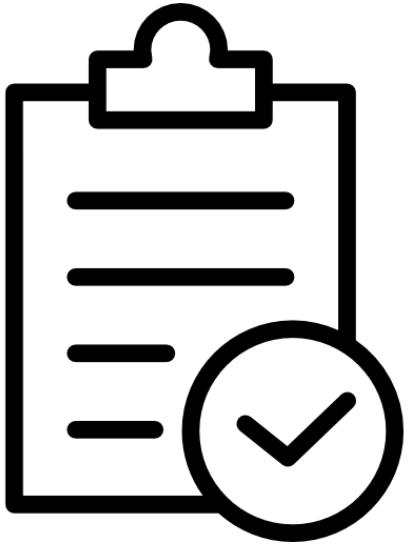
Prof.ssa Chiara Petrioli

SAPIENZA
UNIVERSITÀ DI ROMA

Corso di C

Christian Cardia, Gabriele Saturni

Esercitazione



Esercizio 1

Si scriva una funzione con il seguente prototipo:

*int fun(int arr[ ], int len, int *diff)*

che prende come parametri un array di interi, la sua dimensione e un puntatore a intero. La funzione deve ritornare la somma di tutti gli elementi dell'array e deve salvare in **diff,* la differenza tra l'elemento maggiore ed il minore.

Soluzione esercizio 1 (1/2)

```c
#include<stdio.h>

//prototipo funzione
int fun(int arr[], int len, int *diff);

int main(){

    int size = 10;
    int array[size];

    for(int i=0; i<size;i++){
        printf("%d) Inserisci numero: ",i+1);
        scanf("%d",&array[i]);
        printf("\n");
    }

    int diff;

    int somma = fun(array, size, &diff);

    printf("\n Somma: %d, differenza: %d \n",somma,diff);

    return 0;

}
```

Soluzione esercizio 1 (2/2)

```
38 int fun(int arr[], int len, int *diff){
39
40     if(len == 0){
41         *diff = 0;
42         return 0;
43     }
44
45     int min = arr[0];
46     int max = arr[0];
47     int somma = arr[0];
48
49     for(int i=1; i<len; i++){
50         somma += arr[i];
51         if(arr[i] < min){
52             min = arr[i];
53         }
54
55         if(arr[i] > max){
56             max = arr[i];
57         }
58     }
59
60     *diff = max - min;
61     return somma;
62
63 }
```

Esercizio 2

Si scriva una funzione con il seguente prototipo:

*int *insertElements(int *size)*

La funzione deve permettere all'utente di inserire numeri interi fino a che non inserisce *-1,* salvando i numeri su un array dinamico. La funzione deve ritornare l'indirizzo del primo elemento dell'array e salvare nel puntatore **size* la sua dimensione.

Dal main, dopo aver chiamato la funzione, si stampino tutti gli elementi dell'array.

Soluzione esercizio 2 (1/2)

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

//prototipo funzione
int* insertElements(int *size);

int main(){

    int size = 0;
    int *ptr = insertElements(&size);

    if((size==0) || (ptr==NULL)){
        printf("\nNessun elemento ! \n");
        return 0;
    }

    printf("\nElementi inseriti -->");
    for(int i=0;i<size;i++){
        printf("\n%d) %d",i+1,ptr[i]);
    }

    free(ptr);

    printf("\n");

    return 0;

}
```

Soluzione esercizio 2 (2/2)

```c
int* insertElements(int *size){

    int *array = NULL;
    *size = 0;

    while(1){
        int input = -1;
        printf("\nInserire un intero oppure '-1' per terminare: ");
        scanf("%d",&input);
        if(input == -1){
            break;
        }
        *size += 1;
        array = (int *)realloc(array, *size * sizeof(int));
        if(array == NULL){
            printf("\nErrore allocazione");
            *size = 0;
            return array;
        }
        array[*size-1] = input;
    }

    return array;
}
```

Esercizio 3

Si scriva cosa stampano le 5 printf:

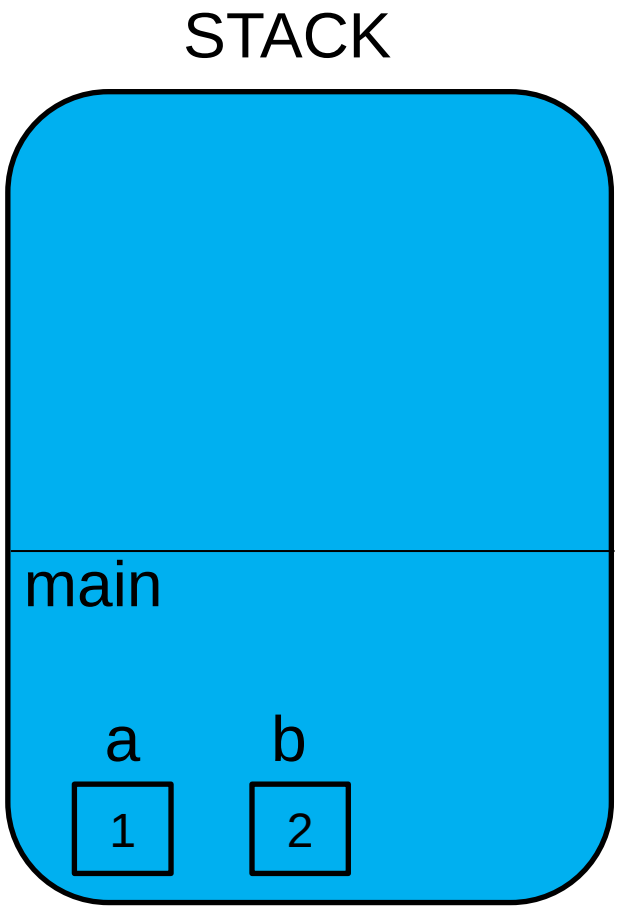
```c
void funzione1(int *num1, int num2);
void funzione2(int num1, int *num2);

int main(){

    int a = 1;
    int b = 2;
    printf("main1 --> a: %d, b: %d",a,b);
    funzione1(&a,b);
    printf("\nmain2 --> a: %d, b: %d",a,b);
    funzione2(a,&b);
    printf("\nmain3 --> a: %d, b: %d \n",a,b);

    return 0;

}

void funzione1(int *num1, int num2){
    num2 = *num1 + 3;
    *num1 = *num1 + 12;
    printf("\nfunzione1 --> num1: %d, num2: %d",*num1,num2);
}

void funzione2(int num1, int *num2){
   *num2 = num1;
   printf("\nfunzione2 --> num1: %d, num2: %d",num1,*num2);
}
```

Soluzione esercizio 3

```
main1 --> a: 1, b: 2
funzione1 --> num1: 13, num2: 4
main2 --> a: 13, b: 2
funzione2 --> num1: 13, num2: 13
main3 --> a: 13, b: 13
```

Soluzione esercizio 3

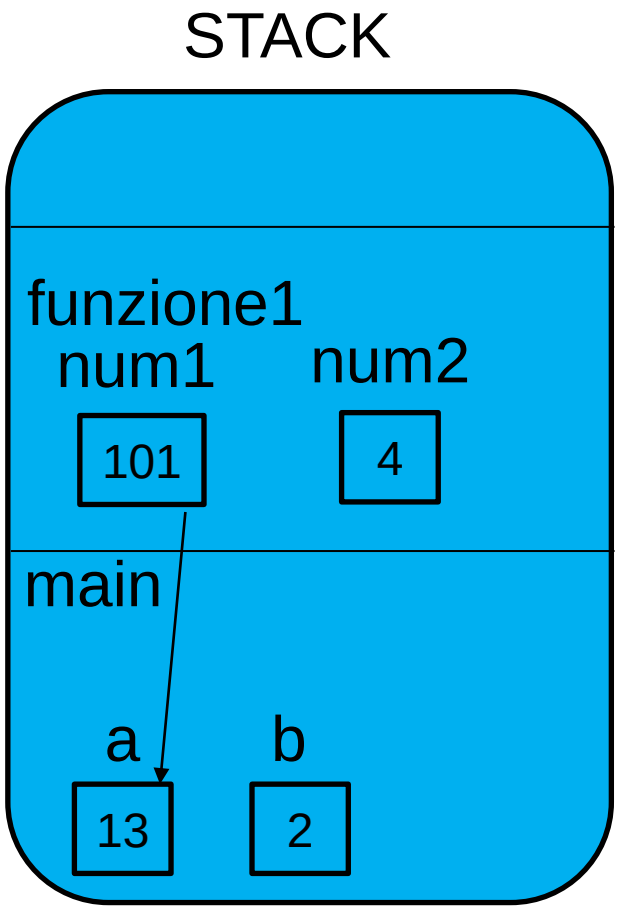
```c
void funzione1(int *num1, int num2);
void funzione2(int num1, int *num2);

int main(){

    int a = 1;
    int b = 2;
    printf("main1 --> a: %d, b: %d",a,b);
    funzione1(&a,b);
    printf("\nmain2 --> a: %d, b: %d",a,b);
    funzione2(a,&b);
    printf("\nmain3 --> a: %d, b: %d \n",a,b);

    return 0;

}

void funzione1(int *num1, int num2){
    num2 = *num1 + 3;
    *num1 = *num1 + 12;
    printf("\nfunzione1 --> num1: %d, num2: %d",*num1,num2);
}

void funzione2(int num1, int *num2){
   *num2 = num1;
   printf("\nfunzione2 --> num1: %d, num2: %d",num1,*num2);
}
```



Main1 → a: 1, b: 2

Soluzione esercizio 3

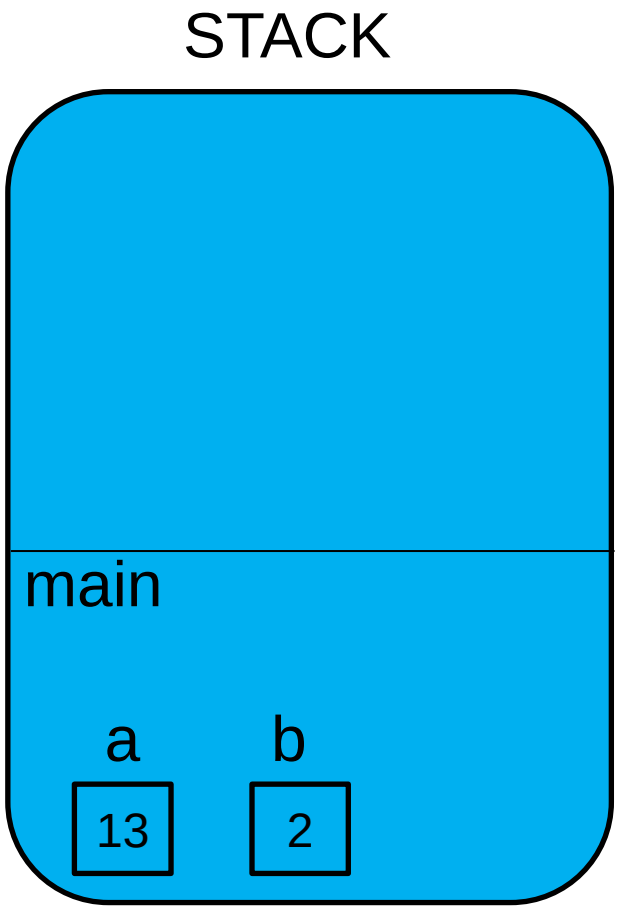
```c
void funzione1(int *num1, int num2);
void funzione2(int num1, int *num2);

int main(){

    int a = 1;
    int b = 2;
    printf("main1 --> a: %d, b: %d",a,b);
    funzione1(&a,b);
    printf("\nmain2 --> a: %d, b: %d",a,b);
    funzione2(a,&b);
    printf("\nmain3 --> a: %d, b: %d \n",a,b);

    return 0;

}

void funzione1(int *num1, int num2){
    num2 = *num1 + 3;
    *num1 = *num1 + 12;
    printf("\nfunzione1 --> num1: %d, num2: %d",*num1,num2);
}

void funzione2(int num1, int *num2){
   *num2 = num1;
    printf("\nfunzione2 --> num1: %d, num2: %d",num1,*num2);
}
```



funzione1 → num1: 13, num2: 4

Soluzione esercizio 3

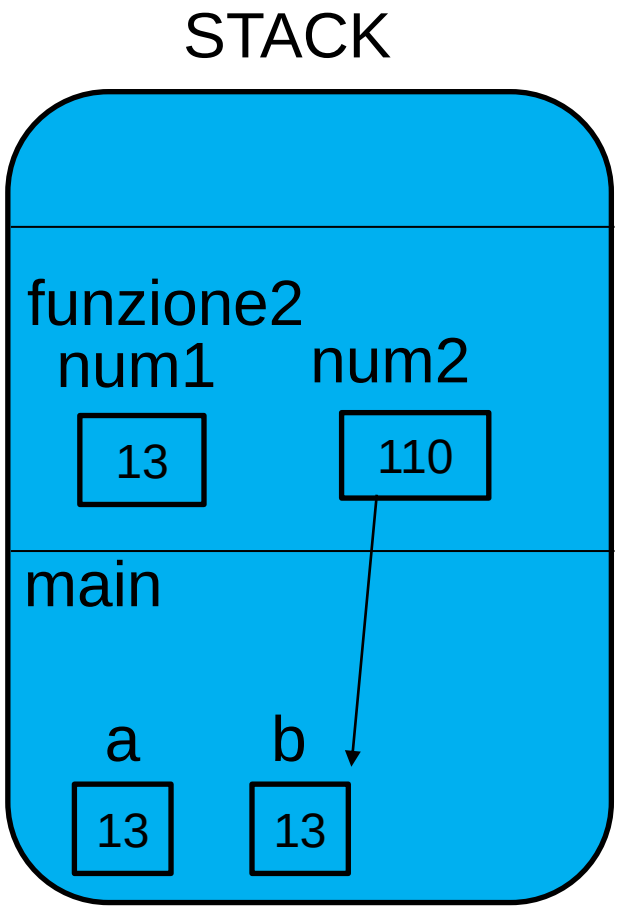
```c
void funzione1(int *num1, int num2);
void funzione2(int num1, int *num2);

int main(){

    int a = 1;
    int b = 2;
    printf("main1 --> a: %d, b: %d",a,b);
    funzione1(&a,b);
    printf("\nmain2 --> a: %d, b: %d",a,b);
    funzione2(a,&b);
    printf("\nmain3 --> a: %d, b: %d \n",a,b);

    return 0;

}

void funzione1(int *num1, int num2){
    num2 = *num1 + 3;
    *num1 = *num1 + 12;
    printf("\nfunzione1 --> num1: %d, num2: %d",*num1,num2);
}

void funzione2(int num1, int *num2){
   *num2 = num1;
   printf("\nfunzione2 --> num1: %d, num2: %d",num1,*num2);
}
```



main2 → a: 13, b: 2

Soluzione esercizio 3

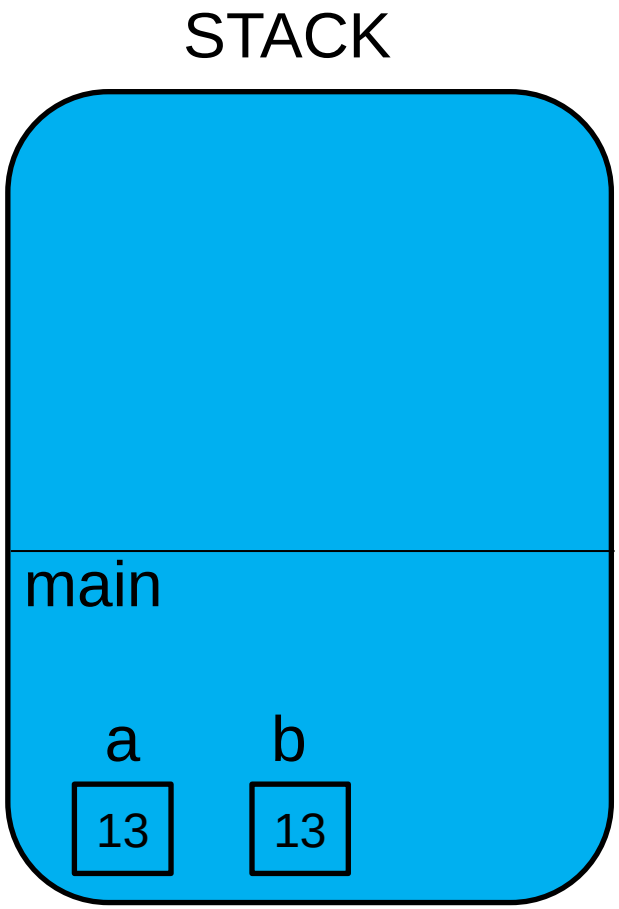
```c
void funzione1(int *num1, int num2);
void funzione2(int num1, int *num2);

int main(){

    int a = 1;
    int b = 2;
    printf("main1 --> a: %d, b: %d",a,b);
    funzione1(&a,b);
    printf("\nmain2 --> a: %d, b: %d",a,b);
    funzione2(a,&b);
    printf("\nmain3 --> a: %d, b: %d \n",a,b);

    return 0;

}

void funzione1(int *num1, int num2){
    num2 = *num1 + 3;
    *num1 = *num1 + 12;
    printf("\nfunzione1 --> num1: %d, num2: %d",*num1,num2);
}

void funzione2(int num1, int *num2){
   *num2 = num1;
   printf("\nfunzione2 --> num1: %d, num2: %d",num1,*num2);
}
```



Funzione2 → num1: 13, num2: 13

Soluzione esercizio 3

```c
void funzione1(int *num1, int num2);
void funzione2(int num1, int *num2);

int main(){

    int a = 1;
    int b = 2;
    printf("main1 --> a: %d, b: %d",a,b);
    funzione1(&a,b);
    printf("\nmain2 --> a: %d, b: %d",a,b);
    funzione2(a,&b);
    printf("\nmain3 --> a: %d, b: %d \n",a,b);

    return 0;

}

void funzione1(int *num1, int num2){
    num2 = *num1 + 3;
    *num1 = *num1 + 12;
    printf("\nfunzione1 --> num1: %d, num2: %d",*num1,num2);
}

void funzione2(int num1, int *num2){
   *num2 = num1;
    printf("\nfunzione2 --> num1: %d, num2: %d",num1,*num2);
}
```



Main3 → a: 13, b: 13

Esercizio 4

Si scriva un programma che consenta all'utente di inserire numeri interi fino a che non ne inserisce uno negativo. Il programma deve stampare i numeri inseriti dall'utente nell'ordine inverso, la lista dei numeri pari e la lista dei numeri dispari.

Soluzione esercizio 4 (1/3)

```c
int* inserisciElementi(int *size);
void invertiNumeri(int *ptr, int size);
void stampaNumeri(int *ptr, int size);
void stampaPariDispari(int *ptr, int size);

int main(){

    int size = 0;
    int *ptr = inserisciElementi(&size);

    printf("\nNumeri inseriti-->\n");
    stampaNumeri(ptr,size);
    printf("\n");
    stampaPariDispari(ptr,size);
    invertiNumeri(ptr,size);
    printf("\nNumeri invertiti-->\n");
    stampaNumeri(ptr,size);
    printf("\n");

    free(ptr);

    return 0;
}
```

Soluzione esercizio 4 (2/3)

```c
int* inserisciElementi(int *size){
    *size = 0;
    int *ptr = NULL;

    while(1){
        int input;
        printf("\nInserisci un numero: ");
        scanf("%d",&input);

        if(input<0){
            printf("\nFine inserimento...");
            return ptr;
        }

        *size += 1;
        ptr = (int *) realloc(ptr, *size * sizeof(int));
        ptr[*size-1] = input;

    }

    return ptr;

}

void invertiNumeri(int *ptr, int size){
    int a = 0;
    for(int i=(size-1); i>(size-1)/2;i--){
        int num = ptr[i];
        ptr[i] = ptr[a];
        ptr[a] = num;
        a++;
    }

}
```

Soluzione esercizio 4 (3/3)

```
77  void stampaPariDispari(int *ptr, int size){
78
79      printf("\nNumeri pari-->\n");
80      for(int i=0; i<size;i++){
81          if((ptr[i] % 2)==0){
82              printf("%d-",ptr[i]);
83          }
84      }
85
86      printf("\nNumeri dispari-->\n");
87      for(int i=0; i<size;i++){
88          if((ptr[i] % 2)!=0){
89              printf("%d-",ptr[i]);
90          }
91      }
92
93  }
```

Esercizio 5

Si scriva un programma che consenta all'utente di inserire una stringa salvata in un array dinamico.

Il programma deve stampare la stringa invertita.

Soluzione esercizio 5 (1/2)

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

char *inserisciStringa();
void invertiStringa(char *stringa);

int main(){

    char *stringa = inserisciStringa();

    printf("\nStringa inserita: %s",stringa);
    invertiStringa(stringa);
    printf("\nStringa invertita: %s \n",stringa);

    free(stringa);

    return 0;
}
```

Soluzione esercizio 5 (2/2)

```c
char *inserisciStringa(){

    int n = 0;

    char ch;
    char *stringa;

    while((ch=getchar())!= '\n' ){
        n++;
        stringa=(char *) realloc(stringa,n+1);
        if(stringa == NULL){
            printf("Non è possibile allocare spazio..\n");
            return stringa;
        }
        stringa[n-1]=ch;
    }

    stringa[n]= '\0';

    return stringa;

}

void invertiStringa(char *stringa){

    int size = strlen(stringa);
    int a = 0;
    for(int i=(size-1); i>((size-1)/2); i--){
        char c = stringa[a];
        stringa[a] = stringa[i];
        stringa[i] = c;
        a++;
    }

}
```

Esercizio 6

Si scriva cosa stampano le 5 printf:

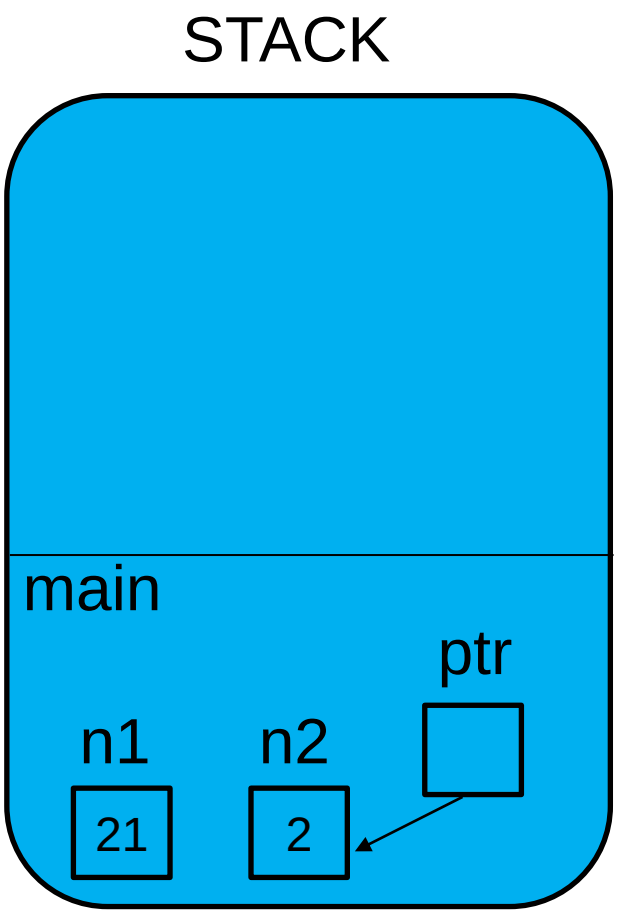
```c
#include<stdio.h>

void f1(int *num1, int num2);
void f2(int num1, int *num2);

int main(){

    int n1 = 21;
    int n2 = 2;
    int *ptr;
    ptr = &n2;
    printf("main1 --> n1: %d, n2: %d, ptr: %d",n1,n2,*ptr);
    f1(&n1,*ptr);
    printf("\nmain2 --> n1: %d, n2: %d, ptr: %d",n1,n2,*ptr);
    f2(n1,&n2);
    printf("\nmain3 --> n1: %d, n2: %d, ptr: %d \n",n1,n2,*ptr);

    return 0;

}

void f1(int *a, int b){
    *a = 32;
    b = *a + 2;
    printf("\nf1 --> a: %d, b: %d",*a,b);
}
void f2(int a, int *b){
   *b = a + 30;
   printf("\nf2 --> a: %d, b: %d",a,*b);
}
```

Soluzione esercizio 6

```
main1 --> n1: 21, n2: 2, ptr: 2
f1 --> a: 32, b: 34
main2 --> n1: 32, n2: 2, ptr: 2
f2 --> a: 32, b: 62
main3 --> n1: 32, n2: 62, ptr: 62
```

Soluzione esercizio 6

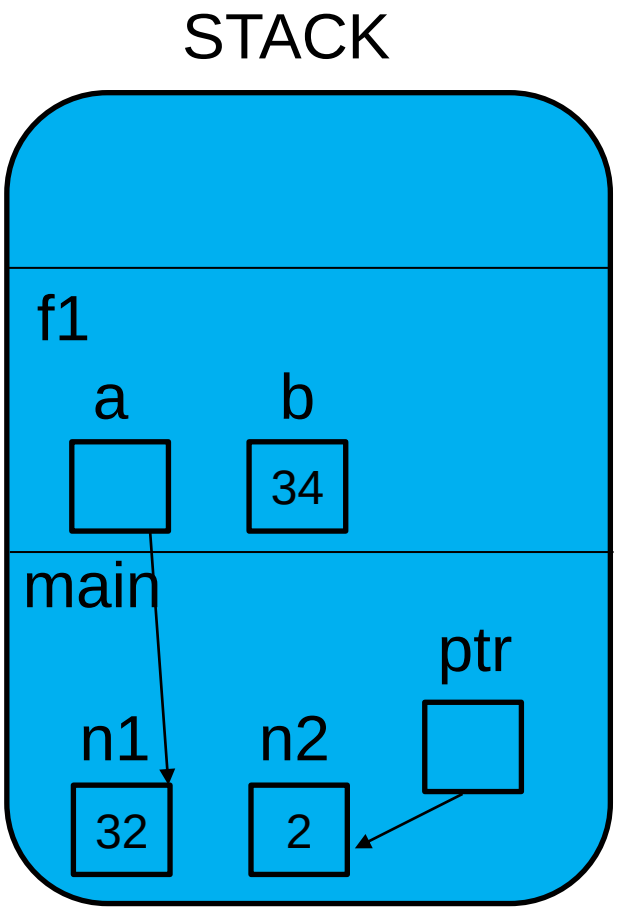
```c
#include<stdio.h>

void f1(int *num1, int num2);
void f2(int num1, int *num2);

int main(){

    int n1 = 21;
    int n2 = 2;
    int *ptr;
    ptr = &n2;
    printf("main1 --> n1: %d, n2: %d, ptr: %d",n1,n2,*ptr);
    f1(&n1,*ptr);
    printf("\nmain2 --> n1: %d, n2: %d, ptr: %d",n1,n2,*ptr);
    f2(n1,&n2);
    printf("\nmain3 --> n1: %d, n2: %d, ptr: %d \n",n1,n2,*ptr);

    return 0;

}

void f1(int *a, int b){
    *a = 32;
    b = *a + 2;
    printf("\nf1 --> a: %d, b: %d",*a,b);
}
void f2(int a, int *b){
   *b = a + 30;
   printf("\nf2 --> a: %d, b: %d",a,*b);
}
```



main1 → n1: 21, n2: 2, ptr: 2

Soluzione esercizio 6

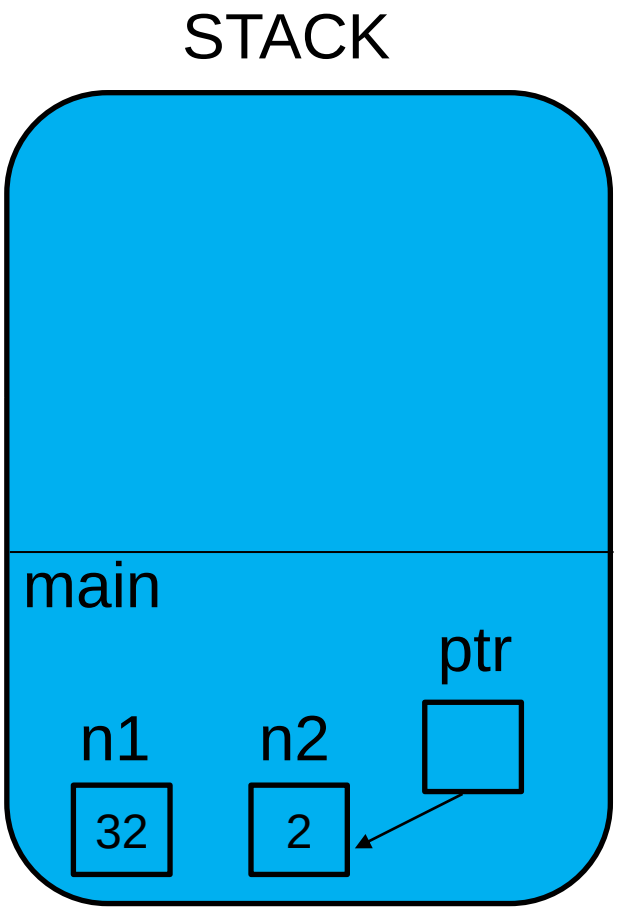
```c
#include<stdio.h>

void f1(int *num1, int num2);
void f2(int num1, int *num2);

int main(){

    int n1 = 21;
    int n2 = 2;
    int *ptr;
    ptr = &n2;
    printf("main1 --> n1: %d, n2: %d, ptr: %d",n1,n2,*ptr);
    f1(&n1,*ptr);
    printf("\nmain2 --> n1: %d, n2: %d, ptr: %d",n1,n2,*ptr);
    f2(n1,&n2);
    printf("\nmain3 --> n1: %d, n2: %d, ptr: %d \n",n1,n2,*ptr);

    return 0;

}

void f1(int *a, int b){
    *a = 32;
    b = *a + 2;
    printf("\nf1 --> a: %d, b: %d",*a,b);
}
void f2(int a, int *b){
   *b = a + 30;
   printf("\nf2 --> a: %d, b: %d",a,*b);
}
```



f1 → a: 32, b: 34

Soluzione esercizio 6

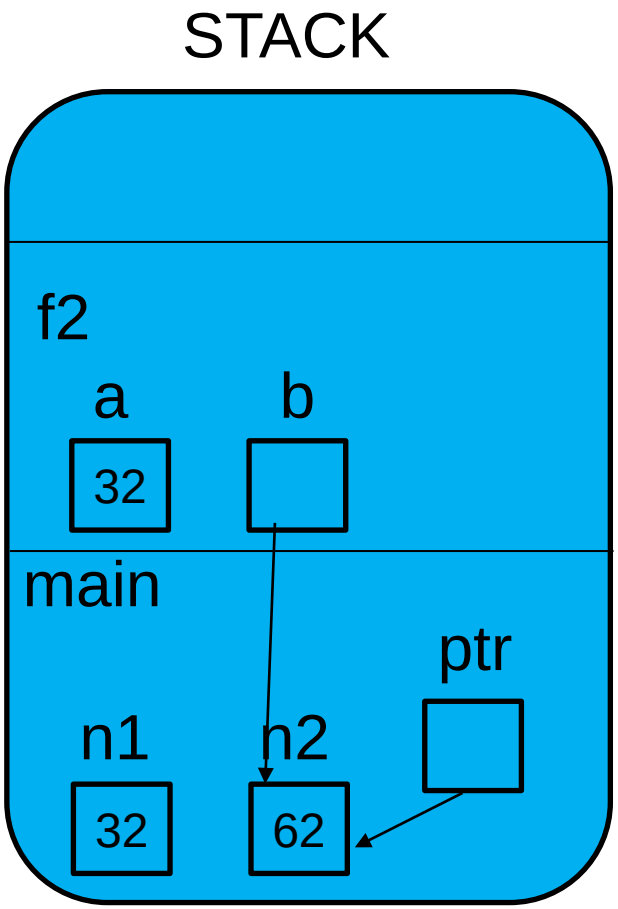
```c
#include<stdio.h>

void f1(int *num1, int num2);
void f2(int num1, int *num2);

int main(){

    int n1 = 21;
    int n2 = 2;
    int *ptr;
    ptr = &n2;
    printf("main1 --> n1: %d, n2: %d, ptr: %d",n1,n2,*ptr);
    f1(&n1,*ptr);
    printf("\nmain2 --> n1: %d, n2: %d, ptr: %d",n1,n2,*ptr);
    f2(n1,&n2);
    printf("\nmain3 --> n1: %d, n2: %d, ptr: %d \n",n1,n2,*ptr);

    return 0;

}

void f1(int *a, int b){
    *a = 32;
    b = *a + 2;
    printf("\nf1 --> a: %d, b: %d",*a,b);
}
void f2(int a, int *b){
   *b = a + 30;
   printf("\nf2 --> a: %d, b: %d",a,*b);
}
```



main2 → n1: 32, n2: 2, ptr: 2

Soluzione esercizio 6

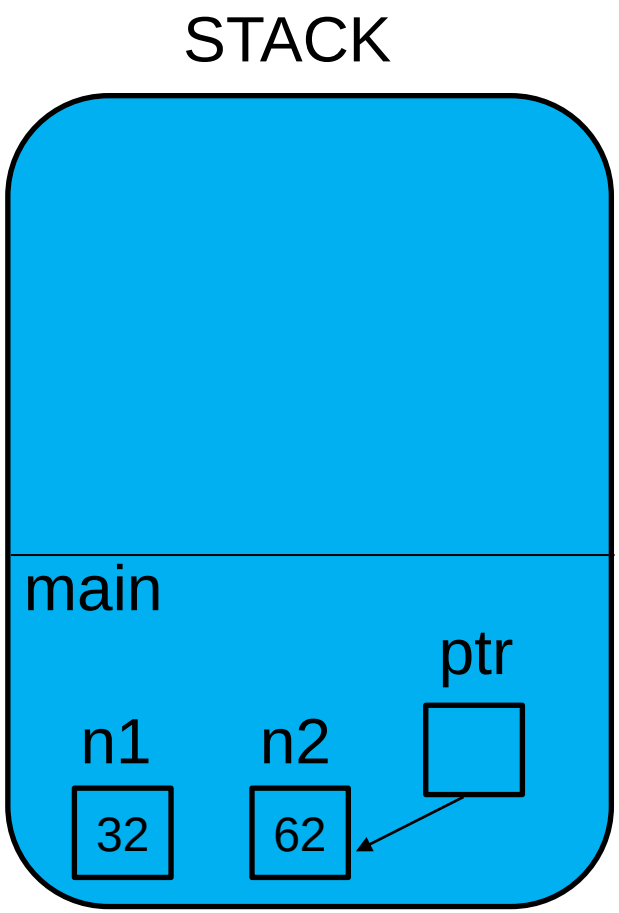
```c
#include<stdio.h>

void f1(int *num1, int num2);
void f2(int num1, int *num2);

int main(){

    int n1 = 21;
    int n2 = 2;
    int *ptr;
    ptr = &n2;
    printf("main1 --> n1: %d, n2: %d, ptr: %d",n1,n2,*ptr);
    f1(&n1,*ptr);
    printf("\nmain2 --> n1: %d, n2: %d, ptr: %d",n1,n2,*ptr);
    f2(n1,&n2);
    printf("\nmain3 --> n1: %d, n2: %d, ptr: %d \n",n1,n2,*ptr);

    return 0;

}

void f1(int *a, int b){
    *a = 32;
    b = *a + 2;
    printf("\nf1 --> a: %d, b: %d",*a,b);
}
void f2(int a, int *b){
   *b = a + 30;
   printf("\nf2 --> a: %d, b: %d",a,*b);
}
```



f2 → a: 32, b: 62

Soluzione esercizio 6

```c
#include<stdio.h>

void f1(int *num1, int num2);
void f2(int num1, int *num2);

int main(){

    int n1 = 21;
    int n2 = 2;
    int *ptr;
    ptr = &n2;
    printf("main1 --> n1: %d, n2: %d, ptr: %d",n1,n2,*ptr);
    f1(&n1,*ptr);
    printf("\nmain2 --> n1: %d, n2: %d, ptr: %d",n1,n2,*ptr);
    f2(n1,&n2);
    printf("\nmain3 --> n1: %d, n2: %d, ptr: %d \n",n1,n2,*ptr);

    return 0;

}

void f1(int *a, int b){
    *a = 32;
    b = *a + 2;
    printf("\nf1 --> a: %d, b: %d",*a,b);
}
void f2(int a, int *b){
   *b = a + 30;
   printf("\nf2 --> a: %d, b: %d",a,*b);
}
```



main3 → n1: 32, n2: 62, ptr: 62

Esercizio 7

Si scriva un programma che dichiara un array di 10 interi da far inserire all'utente. Si scriva una funzione con prototipo:

int funzione(int arr[ ], int size,int numero, int nuovoNumero)

La funzione deve sostituire nell'array tutte le occorrenze di «numero» con «nuovoNumero» e restituire la prima posizione in cui trova «numero».

Se «numero» non è presente, deve restituire -1

Soluzione esercizio 7 (1/2)

```c
#include<stdio.h>

int funzione(int arr[], int size, int numero, int nuovoNumero);

int main(){

    int size = 10;
    int array[10];
    int numeroTrovare;
    int numeroSostituire;

    for(int i=0;i<size;i++){
        printf("\n Inserisci il numero in posizione %d: ",i+1);
        scanf("%d",&array[i]);
    }

    printf("\nInserisci il numero da trovare: ");
    scanf("%d",&numeroTrovare);
    printf("\nInserisci il numero da sostituire: ");
    scanf("%d",&numeroSostituire);

    int prima_occorrenza = funzione(array, size, numeroTrovare, numeroSostituire);

    printf("\nPrima occorrenza: %d",prima_occorrenza);

    printf("\nArray modificato-->");

    for(int i=0;i<size;i++){
        printf("%d-",array[i]);
    }

    printf("\n");

    return 0;
```

Soluzione esercizio 7 (2/2)

```
48  int funzione(int arr[], int size, int numero, int nuovoNumero){
49
50      int prima_occorrenza = -1;
51      for(int i=0;i<size;i++){
52          if(arr[i] == numero){
53              arr[i] = nuovoNumero;
54              if(prima_occorrenza == -1){
55                  prima_occorrenza = i;
56              }
57          }
58      }
59
60      return prima_occorrenza;
61  }
```