

Es. 8.1 – Tutorial n.1 sull'uso del debugger

Si consideri il seguente programma C++. Eseguire passo per passo le istruzioni del programma mediante l'utilizzo del debugger. Ispezionare il valore di tutte le variabili che compaiono nel programma.

```
#include <iostream>
using namespace std;

int f(int a)
{
    a *= 2;
    return a;
}

int main()
{
    int n;
    n=2;

    int*p;
    p = &n;
    *p = 5;

    int& r=n;
    r = 9;

    int m = f(n);

    cout << n << " " << m << endl;

    return 0;
}
```

Es. 8.2 – Tutorial n.2 sull'uso del debugger

Si consideri il seguente programma C++. Eseguire passo per passo le istruzioni del programma mediante l'utilizzo del debugger. Ispezionare il valore di tutte le variabili che compaiono nel programma e individuare eventuali errori.

```
#include <iostream>
#include <cstring>
using namespace std;

struct punto
{
    float x;
    float y;
};

int main ()
{
    // definizione di due elementi di tipo "punto"
    punto p1;
    p1.x=3.3;
    p1.y=1.1;

    punto p2;
    p2=p1;

    // definizione di un array di 3 interi
    int v[3];
    int i=0;
    while (i<3)
    {
        v[i] = i;
        i++;
    }
    v[4]=123;

    // definizione di una stringa
    char w[5];
    strcpy(w, "si");
    strcpy(w, "informatica");

    // stampa della stringa
    cout << w << endl;

    return 0;
}
```

Es. 8.3 – Utilizzo del debugger

Si consideri il seguente programma C++ che:

- legge da tastiera un numero intero n ;
- legge da tastiera n numeri interi e li memorizza in un vettore allocato nello heap;
- calcola il valore massimo tra gli n numeri memorizzati nel vettore;
- somma il massimo così ottenuto a tutti gli elementi del vettore;
- moltiplica tra loro tutti gli elementi del vettore risultante;
- stampa a video il risultato finale;
- dealloca il vettore dallo heap.

```
#include <iostream>
using namespace std;

int main() {
    int n;
    cout << "Inserisci n:" << endl;
    cin >> n;
    if(n < 1) {
        cout << "Numero non valido" << endl;
        return 0;
    }
    int* v = new int[n];

    cout << "Inserisci " << n << " numeri:" << endl;
    for(int i = 1; i <= n; i++)
        cin >> v[i];

    int max = 0;
    for(int i = 1; i <= n; i++)
        max = (v[i-1] > max)?v[i-1]:max;

    for(int i = 1; i <= n; i++)
        v[i-1] += max;

    int mul = v[0];
    for(int i = 1; i <= n; i++)
        mul *= v[i-1];

    cout << "Risultato del calcolo:" << endl;
    cout << mul;

    delete v;

    return 0;
}
```

Tale programma contiene alcuni bug, come dimostrato dal seguente output:

```
Inserisci n:  
3  
Inserisci 3 numeri:  
4  
7  
10  
Risultato del calcolo:  
7546      <- dovrebbe restituire 4760!
```

Il risultato corretto dovrebbe essere 4760, in quanto:

```
max{4, 7, 10} = 10  
{4+10, 7+10, 10+10} = {14, 17, 20}  
14*17*20 = 4760
```

Utilizzare il debugger per trovare e correggere tutti i bug presenti nel programma, in modo da ottenere l'output desiderato.

Es. 8.4 – Liste

PRIMA PARTE:

Scrivere un programma C++ che:

- crea una lista di interi vuota;
- inserisce l'elemento con valore 50 nella lista;
- inserisce l'elemento con valore 60 *in fondo* alla lista;
- inserisce l'elemento con valore 70 *in testa* alla lista.

Utilizzare il debugger per verificare la correttezza del programma.

SECONDA PARTE:

Scrivere una funzione `stampa ()` che:

- prende come argomento una lista di interi;
- mostra a video il valore di tutti i suoi elementi.

Estendere la prima parte dell'esercizio in modo che il programma:

- invochi la funzione `stampa ()` sulla lista appena creata;
- elimini il secondo elemento dalla lista (elemento con valore 50);
- invochi nuovamente la funzione `stampa ()` sulla lista.

Output di esempio:

```
70 50 60
70 60
```

Es. 8.5 – Inserisci elemento in lista ordinata

Scrivere una funzione `inserisciOrdinato(...)` che prende in ingresso una lista di interi `L` (supposta già ordinata in ordine crescente) e un numero intero `n`. La funzione inserisce nella lista un nuovo elemento con valore `n` in una posizione tale da farla rimanere ordinata.

Scrivere un programma C++ che:

- crea una lista vuota;
- legge 6 numeri interi da tastiera;
- li inserisce in modo ordinato nella lista invocando la funzione `inserisciOrdinato(...)`;
- mostra a video la lista risultante.

Output di esempio:

```
Inserisci 6 numeri:
-3
6
4
0
2
-4
Risultato:
-3 -2 0 4 4 6
```