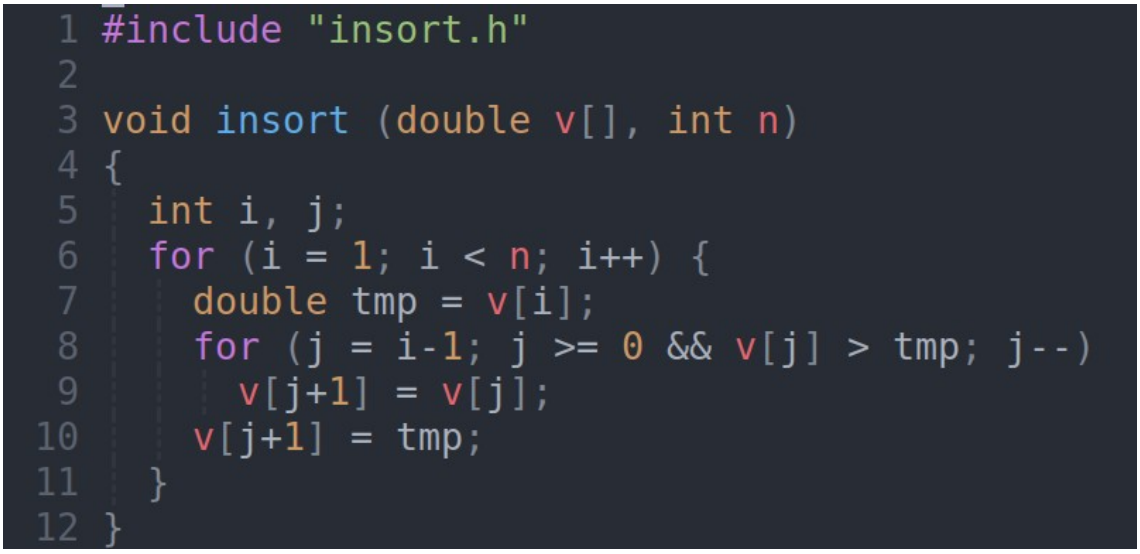
Plik insort.c

Oryginalny kod:

```
#include "insort.h"

void
insort (double v[], int n)
{
  int i, j;
  for (i = 1; i < n; i++) {
    double tmp = v[j];
    for (j = i-1; v[j] > tmp, j--)
      v[j+1] = v[j];
    v[j+1] = tnp;
  }
}
```

Kod poprawiony:



```
1  #include "insort.h"
2
3  void insort (double v[], int n)
4  {
5    int i, j;
6    for (i = 1; i < n; i++) {
7      double tmp = v[i];
8      for (j = i-1; j >= 0 && v[j] > tmp; j--)
9        v[j+1] = v[j];
10     v[j+1] = tmp;
11   }
12 }
```

Zmiany:

- zamieniony przecinek na średnik w linijce 8. Błąd znaleziony podczas kompilacji (wyjście komendy cc zwróciło błąd i wskazało na linijkę)
- zamienione “tnp” na “tmp” w linijce 10. Błąd znaleziony podczas kompilacji (wyjście komendy cc zwróciło błąd i wskazało na linijkę)
- zamienione v[j] na v[i] w linijce 7. Błąd znaleziony po analizie kodu – j w tamtym momencie nie miało ustalonej wartości.
- Dodanie dodatkowego warunku pętli for w linijce 8 (j >= 0). Błąd znaleziony podczas debugowania programem gdb. Zmienna j stawała się ujemna przez co występowało mazanie po pamięci

Plik q_sort.c

Oryginalny kod:

```
int divide( double v[], int f, int l ) {
  double tmp; // zmienna tymczasowa do zmiany elementów
  int s= f;   // dzielimy ze względu na pierwszy element
  f++;        // opuść pierwszy element
  while( f < l ) {
    while( f < l && v[f] < v[s] ) // przeskocz < v[s]
      f++;
    while( f < l && v[l] >= v[s] ) // przeskocz >= v[s]
      l--;
    if( f < l ) { // zamień v[f] i v[l], gdy są różne
      tmp= v[f];
      v[f]= v[l];
      v[l]= tmp;
    }
  }
  // zamieniamy element v[s] z v[f]
  // aby v[s] znalazł się pomiędzy
  // mniejszymi i nie-większymi od niego
  tmp = v[s];
  v[s] = v[f];
  v[f] = tmp;
  return f;
}
```

Kod poprawiony:

```
1  int divide( double v[], int f, int l ) {
2    double tmp; // zmienna tymczasowa do zmiany elementów
3          int s= f;   // dzielimy ze względu na pierwszy element
4          f++;        // opuść pierwszy element
5          while( f < l ) {
6                  while( f < l && v[f] < v[s] )
7                          f++;
8                  while( f < l && v[l] >= v[s] )
9                          l--;
10                 if( f < l ) {
11                         tmp= v[f];
12                         v[f]= v[l];
13                         v[l]= tmp;
14                 }
15         }
16   // zamieniamy element v[s] z v[f]
17   // aby v[s] znalazł się pomiędzy
18   // mniejszymi i nie-większymi od niego
19   if(v[f] < v[s])
20   {
21         tmp = v[f];
22         v[f] = v[s];
23         v[s] = tmp;
24         return f;
25   }
26   else
27   {
28         tmp = v[f-1];
29         v[f-1] = v[s];
30         v[s] = tmp;
31         return f-1;
32   }
33 }
```

Zmiany:

- Błąd występował na poziomie zamiany v[s] z v[f]. Istnieje duże ryzyko, że v[f] jest większe niż v[s] przez co umieszczenie go po stronie wartości mniejszych nie posortuje w prawidłowy sposób wektora. Dodać należy zatem warunek w którym sprawdzamy czy v[f] jest mniejsze niż v[s] I dopiero wtedy zamieniamy je miejscami. W przypadku gdy v[f] jest większe zamieniamy v[s] z wartością stojącą o 1 bliżej początku I zwracamy miejsce z którym zamieniliśmy. Robimy tak ze względu na to, że mamy pewność że wartości stojące dalej początku są większe a te stojące bliżej są mniejsze niż v[s]