


АЛГОРИТМИ ЗА СОРТИРАЊЕ

 **Практична задача :** Споредба на времињата на извршување на неколку алгоритми за сортирање. Тестирањата на сите алгоритми се врз исти генерирани случајни низи со помош на временски функции, и со различен број на елементи. Користени се временски функции за пресметка на времињата на извршување.

```
#include <cstdlib>
#include <iostream>
#include <time.h>
```

```
using namespace std;
```

```
double quick_zbir=0;
double insertion_zbir=0;
double selection_zbir=0;
double bubble_zbir=0;
```

```
void random(long int insertion[],long int selection[],long int
bubble[],long int quick[],int j)
{
int broj;
for(int i=0;i<j;i++){
broj=rand()%j+1;
insertion[i]=broj;
selection[i]=broj;
bubble[i]=broj;
quick[i]=broj;}
}
```

```
int partition(long int input[],long int p,long int r)
{int pivot = input[(r+p)/2];
while ( p < r )
{while ( input[p] < pivot )
p++;
```

```
while ( input[r] > pivot )
r--;
if ( input[p] == input[r] )
p++;
else if ( p < r )
{int tmp = input[p];
input[p] = input[r];
input[r] = tmp;}}
return r;
}
```

```
void quicksort(long int input[], long int p, long int r)
{
if ( p < r )
{int j = partition(input, p, r);
quicksort(input, p, j-1);
quicksort(input, j+1, r);}
}
```

```
void insertionsort(long int a[],long int n)
{
long int j,t;
for(int i=1; i<n; i++){
j=i;
t=a[j];
while (j>0 && a[j-1] > t){
a[j]=a[j-1];
j--;}
a[j]=t;}}
```

```
void selectionsort(long int a[],long int n)
{
long int i,min,j, temp;
for(i=0;i<n-1;i++){
min=i;
for(j=i+1;j<n;j++){
if(a[min]>a[j])
```

```
min=j;}
if (min!=i){
temp=a[i];
a[i]=a[min];
a[min]=temp;}
}}
```

```
void bubblesort(long int a[],long int n)
{
long int i,j, smeni, temp;
for(i=n-1;i>=0;i--){
smeni=0;
for(j=0;j<i;j++){
if(a[j+1]<a[j]){
temp=a[j+1];
a[j+1]=a[j];
a[j]=temp;
smeni++;}}
if (smeni==0){
break;}}
```

```
int main(int argc, char *argv[])
```

```
{
int elementi[]={50,100,250,500,750,1000};// golemina na serii
```

```
int j=0;
```

```
clock_t startTime;
```

```
while(j<6) // elementi[] dimenzija
```

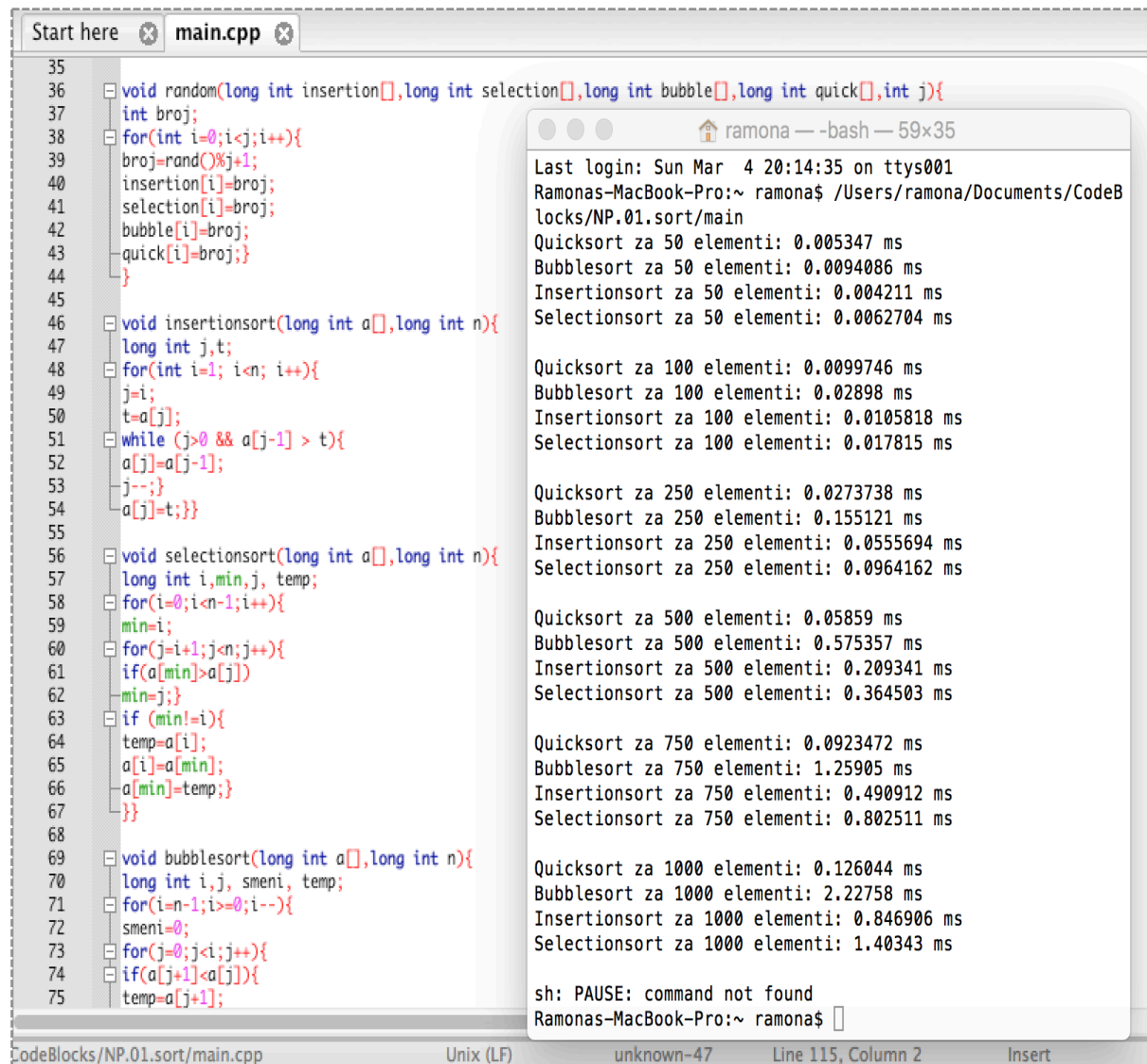
```
{
long int insertion[elementi[j]];
long int selection[elementi[j]];
long int bubble[elementi[j]];
long int quick[elementi[j]];
```

```
for(int povtoruvanja=0;povtoruvanja<5000;povtoruvanja++)
```

```
{  
  
// generiranje na slucajni nizi so 50,100,250,500,750,100 elementi  
  
random(insertion,selection,bubble,quick,elementi[j]);  
startTime = clock();  
quicksort(quick,0, elementi[j]-1);  
quick_zbir=quick_zbir+((double( clock() - startTime ) /  
(double)CLOCKS_PER_SEC)*1000);  
startTime = clock();  
insertionsort(insertion, elementi[j]);  
insertion_zbir=insertion_zbir+((double( clock() - startTime ) /  
(double)CLOCKS_PER_SEC)*1000);  
startTime = clock();  
selectionsort(selection, elementi[j]);  
selection_zbir=selection_zbir+((double( clock() - startTime ) /  
(double)CLOCKS_PER_SEC)*1000);  
startTime = clock();  
bubblesort(bubble, elementi[j]);  
bubble_zbir=bubble_zbir+((double( clock() - startTime ) /  
(double)CLOCKS_PER_SEC)*1000);  
}  
// pecatenje na vreme na sortiranje  
  
cout<<"Quicksort za "<<elementi[j]<<" elementi:  
cout<<"Bubblesort za "<<elementi[j]<<" elementi:  
"<<bubble_zbir/5000<<" ms"<<endl;  
cout<<"Insertionsort za "<<elementi[j]<<" elementi:  
"<<insertion_zbir/5000<<" ms"<<endl;  
cout<<"Selectionsort za "<<elementi[j]<<" elementi:  
"<<selection_zbir/5000<<" ms"<<endl;  
cout<<endl;  
quick_zbir=0; selection_zbir=0; insertion_zbir=0; bubble_zbir=0;  
j++;  
}  
system("PAUSE");  
return EXIT_SUCCESS; }
```

🖥. Извршување и резултати:

Да се тестира кодот, со можност за модификации, да се воочат разликите при промената на бројот на елементи во серијата, и да се направат споредни на резултатите на разни компјутери.



The screenshot shows the CodeBlocks IDE with a C++ file named `main.cpp` open. The code implements four sorting algorithms: `random`, `insertionsort`, `selectionsort`, and `bubblesort`. The `random` function initializes arrays for each algorithm. The other functions perform the respective sorting operations on an array `a` of size `n`.

Overlaid on the IDE is a terminal window titled `ramona — -bash — 59x35`. It shows the output of running the program, displaying execution times in milliseconds for each algorithm with different numbers of elements (50, 100, 250, 500, 750, 1000). The results are as follows:

Algorithm	50 elementi	100 elementi	250 elementi	500 elementi	750 elementi	1000 elementi
Quicksort	0.005347 ms	0.0099746 ms	0.0273738 ms	0.05859 ms	0.0923472 ms	0.126044 ms
Bubblesort	0.0094086 ms	0.02898 ms	0.155121 ms	0.575357 ms	1.25905 ms	2.22758 ms
Insertionsort	0.004211 ms	0.0105818 ms	0.0555694 ms	0.209341 ms	0.490912 ms	0.846906 ms
Selectionsort	0.0062704 ms	0.017815 ms	0.0964162 ms	0.364503 ms	0.802511 ms	1.40343 ms

The terminal also shows a `sh: PAUSE: command not found` error and the prompt `Ramonas-MacBook-Pro:~ ramona$`.