**Insertion sort**

#include<iostream>

#include <iomanip>

#include <cstdlib>

#include <ctime>

#include <ratio>

#include <chrono>

using namespace std;

using namespace std::chrono;

int steps = 0;

void InsertionSort(int A[], int n)

{

int i, value, index;

for (i = 1; i < n; i++)

{

steps += 5;

value = A[i];

index = i;

while (index > 0 && A[index - 1] > value)

{

steps += 4;

A[index] = A[index - 1];

index = index - 1;

}

A[index ] = value;

}

}

int main()

{

srand(0);

int n[8] = { 100, 300, 500, 1000, 3000, 5000, 10000, 50000 };

int\* a1;

int\* a2;

int\* a3;

for (int i = 0; i < 3; i++)

{

//for loop for Task two 50 times (incomplete)

int size = n[i];

a1 = new int[size];

a2 = new int[size];

a3 = new int[size];

//assign 1,2,...n-1, n to a1

for (int j = 1; j <= size; j++)

{

a1[j - 1] = j;

a3[j - 1] = j;

}

//assign n, n-1, ... , 2, 1 to a2

for (int j = size; j > 0; j--)

{

a2[size - j] = j;

}

//random permutations

for (int k = 0; k < size; k++) // 1 2 3 ....100 its not repeated after numbers I randomized

{

int index = rand() % size;

int temp = a3[k];

a3[k] = a3[index];

a3[index] = temp;

}

cout << endl;

for (int a = 0; a < size; a++)

{

cout << a1[a] << " ";

}

cout << endl;

high\_resolution\_clock::time\_point t1 = high\_resolution\_clock::now();

InsertionSort(a1, size);

high\_resolution\_clock::time\_point t2 = high\_resolution\_clock::now();

duration<double> time\_span1 = duration\_cast<duration<double>>(t2 - t1);

std::cout << "It took me " << time\_span1.count() << " seconds.";

cout << endl;

cout << steps << endl;

steps = 0;

for (int a = 0; a < size; a++)

{

cout << a1[a] << " ";

}

cout << endl;

cout << endl;

for (int a = 0; a < size; a++)

{

cout << a2[a] << " ";

}

cout << endl;

high\_resolution\_clock::time\_point t4 = high\_resolution\_clock::now();

InsertionSort(a2, size);

high\_resolution\_clock::time\_point t5 = high\_resolution\_clock::now();

duration<double> time\_span2 = duration\_cast<duration<double>>(t5 - t4);

std::cout << "It took me " << time\_span2.count() << " seconds.";

cout << endl;

cout << steps << endl;

steps = 0;

for (int a = 0; a < size; a++)

{

cout << a2[a] << " ";

}

cout << endl;

cout << endl;

for (int a = 0; a < size; a++)

{

cout << a3[a] << " ";

}

cout << endl;

high\_resolution\_clock::time\_point t6 = high\_resolution\_clock::now();

InsertionSort(a3, size);

high\_resolution\_clock::time\_point t7 = high\_resolution\_clock::now();

duration<double> time\_span3 = duration\_cast<duration<double>>(t7 - t6);

std::cout << "It took me " << time\_span3.count() << " seconds.";

cout << endl;

cout << steps << endl;

steps = 0;

for (int a = 0; a < size; a++)

{

cout << a3[a] << " ";

}

cout << endl;

}

return 0;

}

| Clock time | Sorted | Reversely sorted | Random Permutation |
| --- | --- | --- | --- |
| 100 | 8.99e-07 seconds | 1.9122e-05 seconds | 1.0749e-05 seconds |
| 300 | 1.794e-06 seconds | 0.000181715 seconds | 7.6485e-05 seconds |
| 500 | 3.403e-06 seconds | 0.000559153 seconds | 0.000292767 seconds |
| 1000 | 3.442e-06 seconds | 0.00133436 seconds | 0.000616392 seconds |
| 3000 | 2.549e-05 seconds | 0.0198234 seconds | 0.00692091 seconds |
| 5000 | 1.12e-05 | 0.020247 seconds | 0.0105393 seconds |
| 10000 | 2.4e-05 seconds | 0.0845783 seconds | 0.0444097 seconds |
| 50000 | 0.000155291 | 2.08769 seconds | 1.13713 seconds |

**Selection sort**

#include<iostream>

#include <iomanip>

#include <cstdlib>

#include <ctime>

#include <ratio>

#include <chrono>

using namespace std;

using namespace std::chrono;

int steps = 0;

void swap(int\* xp, int\* yp)

{

int temp = \*xp;

\*xp = \*yp;

\*yp = temp;

}

void SelectionSort(int a[], int n) {

for (int i = 1; i <= n - 1; i++) {

steps += 3;

int large = 0;

for (int j = 1; j <= n - i; j++) {

steps += 2;

if (a[large] < a[j]) {

steps += 3;

large = j;

}

swap(a[large], a[n - i]);

}

}

}

int main()

{

srand(0);

int n[8] = { 100, 300, 500, 1000, 3000, 5000, 10000, 50000 };

int\* a1;

int\* a2;

int\* a3;

for (int i = 0; i < 1; i++)

{

//for loop for Task two 50 times (incomplete)

int size = n[i];

a1 = new int[size];

a2 = new int[size];

a3 = new int[size];

//assign 1,2,...n-1, n to a1

for (int j = 1; j <= size; j++)

{

a1[j - 1] = j;

a3[j - 1] = j;

}

//assign n, n-1, ... , 2, 1 to a2

for (int j = size; j > 0; j--)

{

a2[size - j] = j;

}

//random permutations

for (int k = 0; k < size; k++) // 1 2 3 ....100 its not repeated after numbers I randomized

{

int index = rand() % size;

int temp = a3[k];

a3[k] = a3[index];

a3[index] = temp;

}

cout << endl;

for (int a = 0; a < size; a++)

{

cout << a1[a] << " ";

}

cout << endl;

high\_resolution\_clock::time\_point t1 = high\_resolution\_clock::now();

SelectionSort(a1, size);

high\_resolution\_clock::time\_point t2 = high\_resolution\_clock::now();

duration<double> time\_span1 = duration\_cast<duration<double>>(t2 - t1);

std::cout << "It took me " << time\_span1.count() << " seconds.";

cout << endl;

cout << steps << endl;

steps = 0;

for (int a = 0; a < size; a++)

{

cout << a1[a] << " ";

}

cout << endl;

cout << endl;

for (int a = 0; a < size; a++)

{

cout << a2[a] << " ";

}

cout << endl;

high\_resolution\_clock::time\_point t4 = high\_resolution\_clock::now();

SelectionSort(a2, size);

high\_resolution\_clock::time\_point t5 = high\_resolution\_clock::now();

duration<double> time\_span2 = duration\_cast<duration<double>>(t5 - t4);

std::cout << "It took me " << time\_span2.count() << " seconds.";

cout << endl;

cout << steps << endl;

steps = 0;

for (int a = 0; a < size; a++)

{

cout << a2[a] << " ";

}

cout << endl;

cout << endl;

for (int a = 0; a < size; a++)

{

cout << a3[a] << " ";

}

cout << endl;

high\_resolution\_clock::time\_point t6 = high\_resolution\_clock::now();

SelectionSort(a3, size);

high\_resolution\_clock::time\_point t7 = high\_resolution\_clock::now();

duration<double> time\_span3 = duration\_cast<duration<double>>(t7 - t6);

std::cout << "It took me " << time\_span3.count() << " seconds.";

cout << endl;

cout << steps << endl;

steps = 0;

for (int a = 0; a < size; a++)

{

cout << a3[a] << " ";

}

cout << endl;

}

return 0;

}

| Clock time | Sorted | Reversely sorted | Random Permutation |
| --- | --- | --- | --- |
| 100 | 0.0005482 seconds | 0.0006475 seconds | 0.0006488 seconds. |
| 300 | 0.0047947 seconds | 0.0058907 seconds | 0.0068821 seconds |
| 500 | 0.0146682 seconds | 0.0144309 seconds | 0.0142879 seconds |
| 1000 | 0.0659124 seconds | 0.0768232 seconds | 0.0520661 seconds |
| 3000 | 0.466221 seconds | 0.474491 seconds | 0.539969 seconds |
| 5000 | 1.2925 seconds | 1.31227 seconds | 1.31221 seconds |
| 10000 | 5.12431 seconds | 5.19124 seconds | 5.2702 seconds. |
| 50000 | 2.37237 | 6.90019 seconds | 143.183 seconds. |

**BubbleSort**

#include<iostream>

#include <iomanip>

#include <cstdlib>

#include <ctime>

#include <ratio>

#include <chrono>

using namespace std;

using namespace std::chrono;

int steps = 0;

void swap(int\* xp, int\* yp)

{

int temp = \*xp;

\*xp = \*yp;

\*yp = temp;

}

void BubbleSort(int a[], int n) {

for (int i = 0; i < n - 1; i++)

{

steps += 2;

for (int j = 0; j < n - i - 1; j++) {

steps += 2;

if (a[j] > a[j + 1])

steps += 2;

swap(a[j], a[j + 1]);

}

}

}

int main()

{

srand(0);

int n[8] = {100, 300, 500, 1000, 3000, 5000, 10000, 50000 };

int\* a1;

int\* a2;

int\* a3;

for (int i = 0; i < 2; i++)

{

//for loop for Task two 50 times (incomplete)

int size = n[i];

a1 = new int[size];

a2 = new int[size];

a3 = new int[size];

//assign 1,2,...n-1, n to a1

for (int j = 1; j <= size; j++)

{

a1[j - 1] = j;

a3[j - 1] = j;

}

//assign n, n-1, ... , 2, 1 to a2

for (int j = size; j > 0; j--)

{

a2[size - j] = j;

}

//random permutations

for (int k = 0; k < size; k++) // 1 2 3 ....100 its not repeated after numbers I randomized

{

int index = rand() % size;

int temp = a3[k];

a3[k] = a3[index];

a3[index] = temp;

}

cout << endl;

for (int a = 0; a < size; a++)

{

cout << a1[a] << " ";

}

cout << endl;

high\_resolution\_clock::time\_point t1 = high\_resolution\_clock::now();

BubbleSort(a1, size);

high\_resolution\_clock::time\_point t2 = high\_resolution\_clock::now();

duration<double> time\_span1 = duration\_cast<duration<double>>(t2 - t1);

std::cout << "It took me " << time\_span1.count() << " seconds.";

cout << endl;

cout << steps << endl;

steps = 0;

for (int a = 0; a < size; a++)

{

cout << a1[a] << " ";

}

cout << endl;

cout << endl;

for (int a = 0; a < size; a++)

{

cout << a2[a] << " ";

}

cout << endl;

high\_resolution\_clock::time\_point t4 = high\_resolution\_clock::now();

BubbleSort(a2, size);

high\_resolution\_clock::time\_point t5 = high\_resolution\_clock::now();

duration<double> time\_span2 = duration\_cast<duration<double>>(t5 - t4);

std::cout << "It took me " << time\_span2.count() << " seconds.";

cout << endl;

cout << steps << endl;

steps = 0;

for (int a = 0; a < size; a++)

{

cout << a2[a] << " ";

}

cout << endl;

cout << endl;

for (int a = 0; a < size; a++)

{

cout << a3[a] << " ";

}

cout << endl;

high\_resolution\_clock::time\_point t6 = high\_resolution\_clock::now();

BubbleSort(a3, size);

high\_resolution\_clock::time\_point t7 = high\_resolution\_clock::now();

duration<double> time\_span3 = duration\_cast<duration<double>>(t7 - t6);

std::cout << "It took me " << time\_span3.count() << " seconds.";

cout << endl;

cout << steps << endl;

steps = 0;

for (int a = 0; a < size; a++)

{

cout << a3[a] << " ";

}

cout << endl;

}

return 0;

}

| Clock time | Sorted | Reversely sorted | Random Permutation |
| --- | --- | --- | --- |
| 100 | 1.2875e-05 | 3.5792e-05 | 3.1333e-05 |
| 300 | 0.000110875 | 0.000308208 | 0.000248667 |
| 500 | 0.000331625 | 0.000945959 | 0.000719625 |
| 1000 | 0.00124221 | 0.00376475 | 0.00263154 |
| 3000 | 0.0111079 | 0.0324905 | 0.0254262 |
| 5000 | 0.0318203 | 0.133541 | 0.0960056 |
| 10000 | 0.122187 | 0.336527 | 0.316388 |
| 50000 | 0.35604 | 0.321157 | 2.95033 |

**Merge Sort**

#include<iostream>

#include <iomanip>

#include <cstdlib>

#include <ctime>

#include <ratio>

#include <chrono>

using namespace std;

using namespace std :: chrono;

int steps = 0;

void Merge(int a[], int begin, int mid, int end) {

int i = begin;

int j = mid + 1;

int\* b = new int[end - begin + 1];

int k = 0;

while (i <= mid && j <= end)

{

steps += 2;

if (a[i] <= a[j])

{

steps += 4;

b[k] = a[i];

i++;

k++;

}

else

{

steps += 3;

b[k] = a[j];

j++;

k++;

}

}

while (i <= mid) {

steps += 4;

b[k] = a[i];

k++;

i++;

}

while (j <= end) {

steps += 4;

b[k] = a[j];

k++;

j++;

}

for (i = begin, k = 0; i <= end; i++, k++) {

steps += 4;

a[i] = b[k];

}

}

void Mergesort(int a[], int begin, int end) {

if (begin < end) {

int mid = (begin + end) / 2;

Mergesort(a, begin, mid);

Mergesort(a, mid + 1, end);

Merge(a, begin, mid, end);

}

}

int main()

{

srand(0);

int n[8] = {100, 300, 500, 1000, 3000, 5000, 10000, 50000 };

int\* a1;

int\* a2;

int\* a3;

for (int i = 0; i < 2; i++)

{

//for loop for Task two 50 times (incomplete)

int size = n[i];

a1 = new int[size];

a2 = new int[size];

a3 = new int[size];

//assign 1,2,...n-1, n to a1

for (int j = 1; j <= size; j++)

{

a1[j - 1] = j;

a3[j - 1] = j;

}

//assign n, n-1, ... , 2, 1 to a2

for (int j = size; j > 0; j--)

{

a2[size - j] = j;

}

//random permutations

for (int k = 0; k < size; k++) // 1 2 3 ....100 its not repeated after numbers I randomized

// we choose the number from the array and save it,

{ // then we pick the number from whats left and save it, except number we picked before

// then choose the number again, but without the numbers -

int index = rand() % size; // - we chose before ( last two numbers)

// repeat

// the array is getting shorter and shorter until there only-

int temp = a3[k]; // - one number left to pick

// why ? becuase it need to be same for the all program

a3[k] = a3[index]; // - it cant be different numbers for all programs

// not sure what she meant by all this :c

a3[index] = temp;

}

cout << endl;

for (int a = 0; a < size; a++)

{

cout << a1[a] << " ";

}

cout << endl;

high\_resolution\_clock::time\_point t1 = high\_resolution\_clock::now(); // high resolution clock

Mergesort(a1, 0, size - 1);

high\_resolution\_clock::time\_point t2 = high\_resolution\_clock::now();

duration<double> time\_span1 = duration\_cast<duration<double>>(t2 - t1);

std::cout << "It took me " << time\_span1.count() << " seconds.";

cout << endl;

cout << steps << endl;

steps = 0;

for (int a = 0; a < size; a++)

{

cout << a1[a] << " ";

}

cout << endl;

cout << endl;

for (int a = 0; a < size; a++)

{

cout << a2[a] << " ";

}

cout << endl;

high\_resolution\_clock::time\_point t4 = high\_resolution\_clock::now();

Mergesort(a2, 0, size - 1);

high\_resolution\_clock::time\_point t5 = high\_resolution\_clock::now();

duration<double> time\_span2 = duration\_cast<duration<double>>(t5 - t4);

std::cout << "It took me " << time\_span2.count() << " seconds.";

cout << endl;

cout << steps << endl;

steps = 0;

for (int a = 0; a < size; a++)

{

cout << a2[a] << " ";

}

cout << endl;

cout << endl;

for (int a = 0; a < size; a++)

{

cout << a3[a] << " ";

}

cout << endl;

high\_resolution\_clock::time\_point t6 = high\_resolution\_clock::now();

Mergesort(a3, 0, size - 1);

high\_resolution\_clock::time\_point t7 = high\_resolution\_clock::now();

duration<double> time\_span3 = duration\_cast<duration<double>>(t7 - t6);

std::cout << "It took me " << time\_span3.count() << " seconds.";

cout << endl;

cout << steps << endl;

steps = 0;

for (int a = 0; a < size; a++)

{

cout << a3[a] << " ";

}

cout << endl;

}

return 0;

}

| Clock time | Sorted | Reversely sorted | Random Permutation |
| --- | --- | --- | --- |
| 100 | 2.2e-05 | 9.75e-06 | 1.1917e-05 |
| 300 | 3.7416e-05 | 2.8416e-05 | 4.9125e-05 |
| 500 | 5.1958e-05 | 4.8125e-05 | 6.5e-05 |
| 1000 | 0.000123709 | 0.00010825 | 0.000140166 |
| 3000 | 0.000335583 | 0.000318292 | 0.000469875 |
| 5000 | 0.000596083 | 0.000567833 | 0.000808959 |
| 10000 | 0.00124167 | 0.00122671 | 0.001994 |
| 50000 | 0.00777342 | 0.00697917 | 0.00967238 |

**Quick Sort**

#include<iostream>

#include <iomanip>

#include <cstdlib>

#include <ctime>

#include <ratio>

#include <chrono>

using namespace std;

using namespace std::chrono;

int steps = 0;

int partition(int A[], int begin, int end) {

steps += 2;

int pivot = A[end];

int i = begin - 1;

for (int j = begin; j < end; j++) {

steps += 2;

if (A[j] < pivot) {

steps += 2;

i++;

swap(A[i], A[j]);

}

}

swap(A[i + 1], A[end]);

return i + 1;

}

void QuickSort(int A[], int begin, int end) {

if (begin < end) {

int q = partition(A, begin, end);

QuickSort(A, begin, q - 1);

QuickSort(A, q + 1, end);

}

}

int main()

{

srand(0);

int n[8] = { 100, 300, 500, 1000, 3000, 5000, 10000, 50000 };

int\* a1;

int\* a2;

int\* a3;

for (int i = 0; i < 1; i++)

{

int size = n[i];

a1 = new int[size];

a2 = new int[size];

a3 = new int[size];

//assign 1,2,...n-1, n to a1

for (int j = 1; j <= size; j++)

{

a1[j - 1] = j;

a3[j - 1] = j;

}

//assign n, n-1, ... , 2, 1 to a2

for (int j = size; j > 0; j--)

{

a2[size - j] = j;

}

//random permutations

for (int k = 0; k < size; k++)

{

int index = rand() % size;

int temp = a3[k];

a3[k] = a3[index];

a3[index] = temp;

}

cout << endl;

for (int a = 0; a < size; a++)

{

cout << a1[a] << " ";

}

cout << endl;

high\_resolution\_clock::time\_point t1 = high\_resolution\_clock::now();

QuickSort(a1, 0, size - 1);

high\_resolution\_clock::time\_point t2 = high\_resolution\_clock::now();

duration<double> time\_span1 = duration\_cast<duration<double>>(t2 - t1);

std::cout << "It took me " << time\_span1.count() << " seconds.";

cout << endl;

cout << steps << endl;

steps = 0;

for (int a = 0; a < size; a++)

{

cout << a1[a] << " ";

}

cout << endl;

cout << endl;

for (int a = 0; a < size; a++)

{

cout << a2[a] << " ";

}

cout << endl;

high\_resolution\_clock::time\_point t4 = high\_resolution\_clock::now();

QuickSort(a2, 0, size - 1);

high\_resolution\_clock::time\_point t5 = high\_resolution\_clock::now();

duration<double> time\_span2 = duration\_cast<duration<double>>(t5 - t4);

std::cout << "It took me " << time\_span2.count() << " seconds.";

cout << endl;

cout << steps << endl;

steps = 0;

for (int a = 0; a < size; a++)

{

cout << a2[a] << " ";

}

cout << endl;

cout << endl;

for (int a = 0; a < size; a++)

{

cout << a3[a] << " ";

}

cout << endl;

high\_resolution\_clock::time\_point t6 = high\_resolution\_clock::now();

QuickSort(a3, 0, size - 1);

high\_resolution\_clock::time\_point t7 = high\_resolution\_clock::now();

duration<double> time\_span3 = duration\_cast<duration<double>>(t7 - t6);

std::cout << "It took me " << time\_span3.count() << " seconds.";

cout << endl;

cout << steps << endl;

steps = 0;

for (int a = 0; a < size; a++)

{

cout << a3[a] << " ";

}

cout << endl;

}

return 0;

}

| Clock time | Sorted | Reversely sorted | Random Permutation |
| --- | --- | --- | --- |
| 100 | 8.3207e-05 | 4.9846e-05 | 1.1511e-05 |
| 300 | 0.000704784 | 0.000416522 | 3.9136e-05 |
| 500 | 0.00207392 | 0.00118552 | 6.871e-05 |
| 1000 | 0.0103988 | 0.00464662 | 0.00207392 |
| 3000 | 0.0774238 | 0.0439853 | 0.000621758 |
| 5000 | 0.124212 | 0.0789852 | 0.000837243 |
| 10000 | 0.338597 | 0.220563 | 0.00128958 |
| 50000 | 0.218488 | 0.00127067 | 8.31753 |