Proiect APD

Dominte Mihai

CR3.1B

Introducere

In cadrul acestui proiect am realizat un algoritm de sortare a unui sir de numere folosind metoda Bubble Sort si am studiat performanta de rulare a acestuia pe diferite esantioane de numere.

Algoritmul foloseste o functie de generare a numerelor pentru a constitui vectorul ce trebuie sortat. A fost folosita functia rand() pentru generarea de numere aleatoare.

Functia de sortare prin metoda bubble sort este cronometrata folosind libraria <chrono> rezultatul obtinut find afisat in microsecunde.

Functia de afisare este optionala si afiseaza numerele sortate din vector.

Algoritm, Limbaj, Masina:

-Ca metoda de sortare a fost folosit Bubble Sort, algoritmul complet fiind prezentat mai jos;

-Limbajul folosit a fost C++, codul a fost scris folosind Visual Studio 2019

-Masina pe care a fost rulat foloseste Windows 10, 32GB RAM, procesor AMD Ryzen 2700X cu 8 nuclee

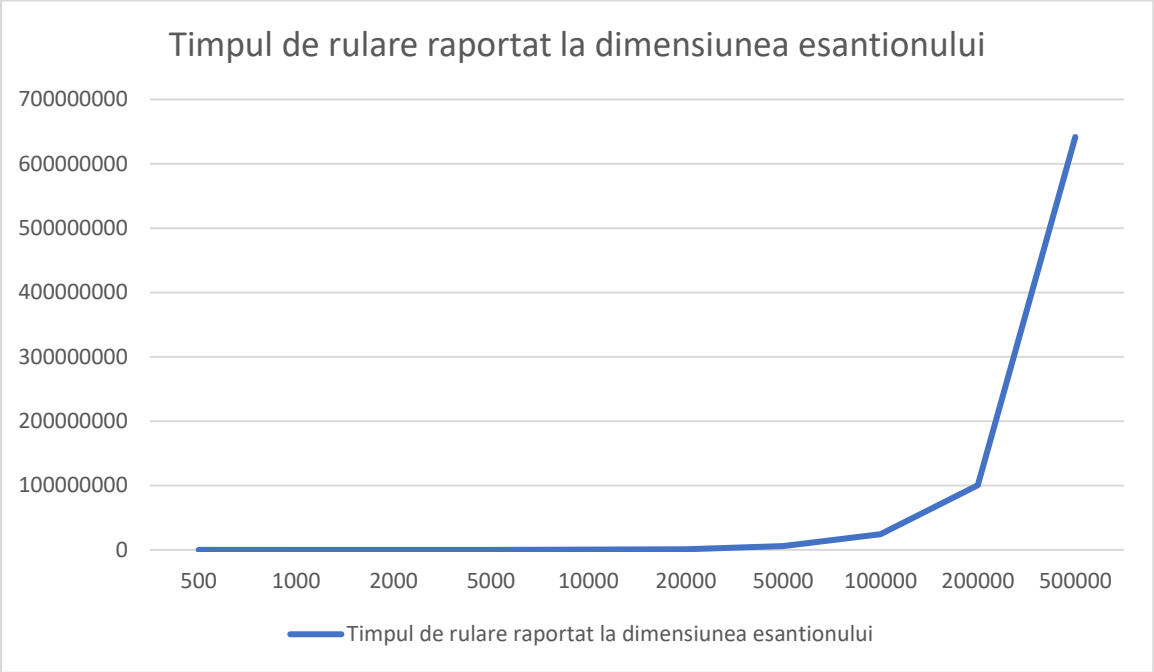
```
#include "stdio.h"
#include <chrono>
#include <iostream>
#include <stdlib.h>
#include <time.h>

using namespace std;
using namespace std::chrono;

void numGenerator(int sir[], int dimensiune);
void bubbleSort(int sir[], int dimensiune);
void afisare(int sir[], int dimensiune);

int main() {
```

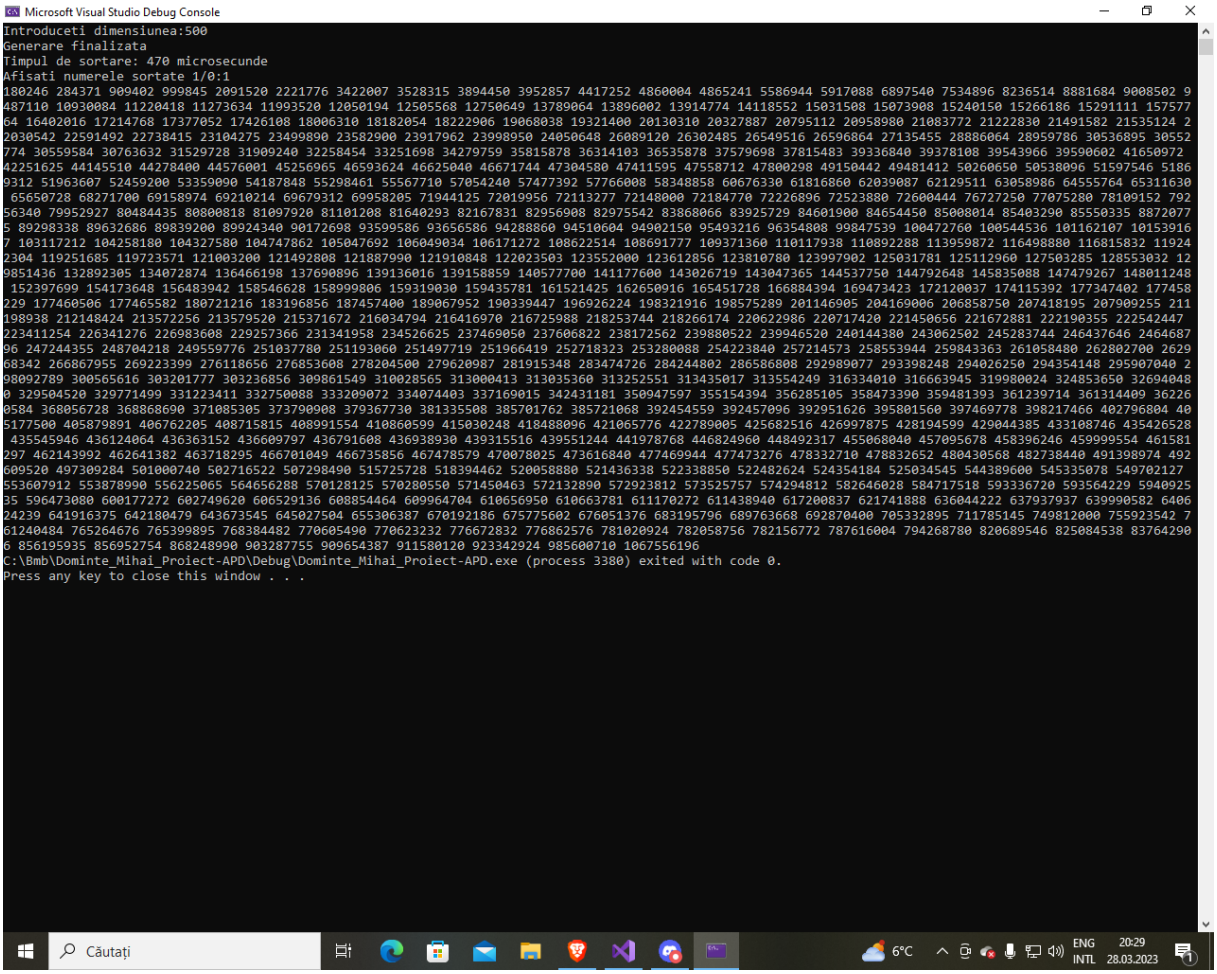
```
	int dimensiune, optiune;

	printf("Introduceti dimensiunea:");
	scanf_s("%d", &dimensiune);

	int* sir = (int*)malloc(dimensiune * sizeof(int));

	numGenerator(sir, dimensiune);
	printf("Generare finalizata \n");

	auto start = high_resolution_clock::now();
	bubbleSort(sir, dimensiune);
	auto stop = high_resolution_clock::now();

	auto duration = duration_cast<microseconds>(stop - start);

	cout << "Timpul de sortare: " << duration.count() << "
microsecunde" << endl;
	printf("Afisati numerele sortate 1/0:");
	scanf_s("%d", &optiune);

	if (optiune == 1) {
		afisare(sir, dimensiune);
	}

	return 0;
}

void numGenerator(int sir[], int dimensiune) {
	srand(time(NULL));
	for (int i = 0; i < dimensiune; i++) {
		sir[i] = rand() * rand();
	}
}

void bubbleSort(int sir[], int dimensiune) {
	for (int i = 0; i < dimensiune - 1; i++) {
		for (int j = 0; j < dimensiune - i - 1; j++) {
			if (sir[j] > sir[j + 1]) {
				int temp = sir[j];
				sir[j] = sir[j + 1];
				sir[j + 1] = temp;
			}
		}
	}
}

void afisare(int sir[], int dimensiune) {
```

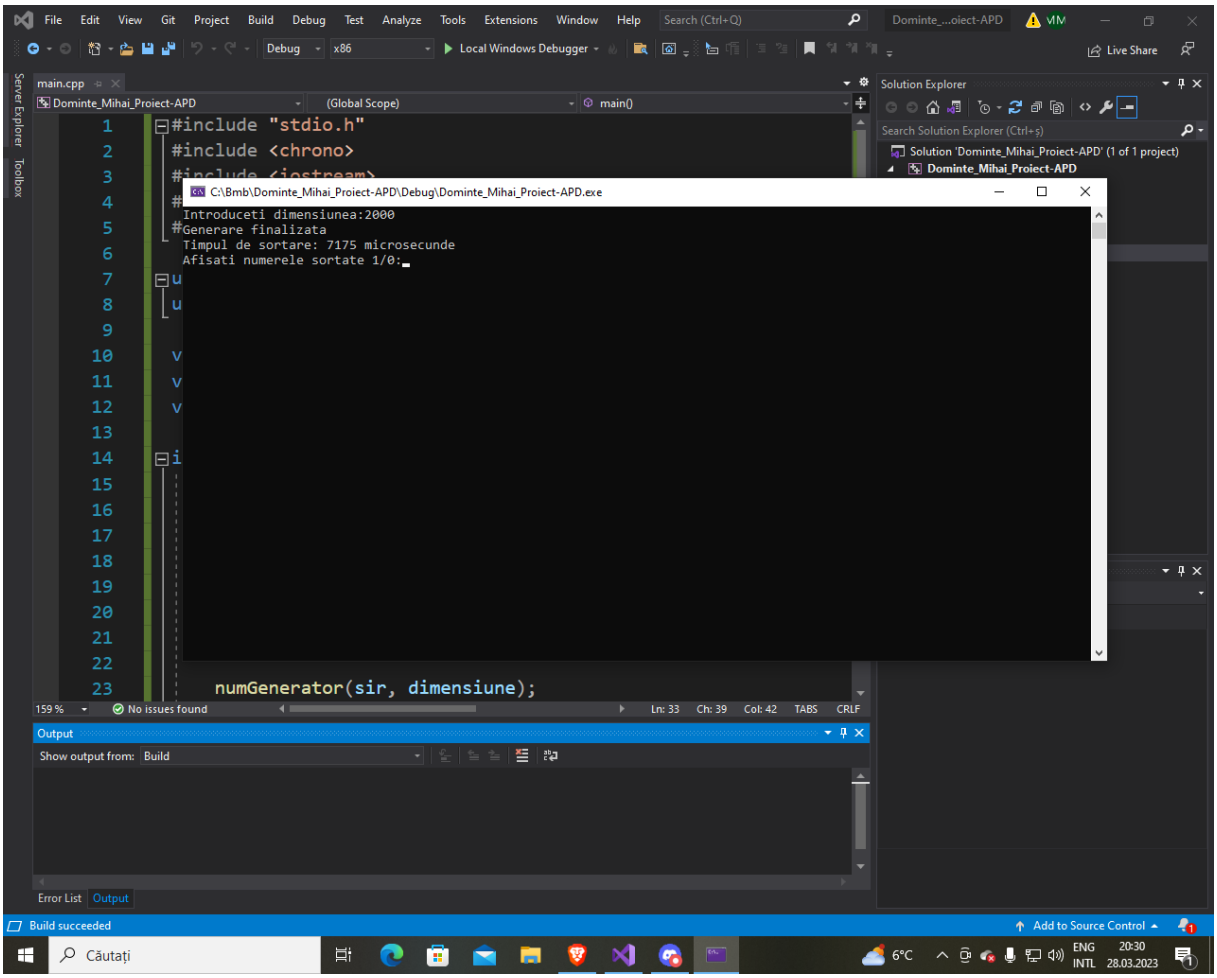
```
	for (int i = 0; i < dimensiune; i++) {
		printf("%d ", sir[i]);
	}
}
```

Teste:

Au fost efectuate 10 teste de sortare cu urmatoarele esantioane de numere: 500, 1000, 2000, 5000, 10000, 20000, 50000, 100000, 200000, 500000 de numere cu rezultatele obtinute prezentate in graficul de mai jos:




Microsoft Visual Studio Debug Console
Introduceti dimensiunea:500
Generare finalizata
Timpul de sortare: 470 microsecunde
Afisati numerele sortate 1/0:1
180246 284371 909402 999845 2091520 2221776 3422007 3528315 3894450 3952857 4417252 4860004 4865241 5586944 5917088 6897540 7534896 8236514 8881684 9008502 9
487110 10930084 11220418 11273634 11993520 12050194 12505568 12750649 13789064 13896002 13914774 14118552 15031508 15073908 15240150 15266186 15291111 157577
64 16402016 17214768 17377052 17426108 18006310 18182054 18222906 19068038 19321400 20130310 20327887 20795112 20958980 21083772 21222830 21491582 21535124 2
2030542 22591492 22738415 23104275 23499890 23582900 23917962 23998950 24050648 26089120 26302485 26549516 26596864 27135455 28886064 28959786 30536895 30552
774 30559584 30763632 31529728 31909240 32258454 33251698 34279759 35815878 36314103 36535878 37579698 37815483 39336840 39378108 39543966 39590602 41650972
42251625 44145510 44278400 44576001 45256965 46593624 46625040 46671744 47304580 47411595 47558712 47800298 49150442 49481412 50260650 50538096 51597546 5186
9312 51963607 52459200 53359090 54187848 55298461 55567710 57054240 57477392 57766008 58348858 60676330 61816860 62039087 62129511 63058986 64555764 65311630
65650728 68271700 69158974 69210214 69679312 69958205 71944125 72019956 72113277 72148000 72184770 72226896 72523880 72600444 76727250 77075280 78109152 792
56340 79952927 80484435 80800818 81097920 81101208 81640293 82167831 82956908 82975542 83868066 83925729 84601900 84654450 85008014 85403290 85550335 8872077
5 89298338 89632686 89839200 89924340 90172698 93599586 93656586 94288860 94510604 94902150 95493216 96354808 99847539 100472760 100544536 101162107 10153916
7 103117212 104258180 104327580 104747862 105047692 106049034 106171272 108622514 108691777 109371360 110117938 110892288 113959872 116498880 116815832 11924
2304 119251685 119723571 121003200 121492808 121887990 121910848 122023503 123552000 123612856 123810780 123997902 125031781 125112960 127503285 128553032 12
9851436 132892305 134072874 136466198 137690896 139136016 139158859 140577700 141177600 143026719 143047365 144537750 144792648 145835088 147479267 148011248
152397699 154173648 156483942 158546628 158999806 159319030 159435781 161521425 162650916 165451728 166884394 169473423 172120037 174115392 177347402 177458
229 177460506 177465582 180721216 183196856 187457400 189067952 190339447 196926224 198321916 198575289 201146905 204169006 206858750 207418195 207909255 211
198938 212148424 213572256 213579520 215371672 216034794 216416970 216725988 218253744 218266174 220622986 220717420 221450656 221672881 222190355 222542447
223411254 226341276 226983608 229257366 231341958 234526625 237469050 237606822 238172562 239880522 239946520 240144380 243062502 245283744 246437646 2464687
96 247244355 248704218 249559776 251037780 251193060 251497719 251966419 252718323 253280088 254223840 257214573 258553944 259843363 261058480 262802700 2629
68342 266867955 269223399 276118656 276853608 278204500 279620987 281915348 283474726 284244802 286586808 292989077 293398248 294026250 294354148 295907040 2
98092789 300565616 303201777 303236856 309861549 310028565 313000413 313035360 313252551 313435017 313554249 316334010 316663945 319980024 324853650 32694048
0 329504520 329771499 331223411 332750088 333209072 334074403 337169015 342431181 350947597 355154394 356285105 358473390 359481393 361239714 361314409 36226
0584 368056728 368868690 371085305 373790908 379367730 381335508 385701762 385721068 392454559 392457096 392951626 395801560 397469778 398217466 402796804 40
5177500 405879891 406762205 408715815 408991554 410860599 415030248 418488096 421065776 422789005 425682516 426997875 428194599 429044385 433108746 435426528
435545946 436124064 436363152 436609797 436791608 436938930 439315516 439551244 441978768 446824960 448492317 455068040 457095678 458396246 459999554 461581
297 462143992 462641382 463718295 466701049 466735856 467478579 470078025 473616840 477469944 477473276 478332710 478832652 480430568 482738440 491398974 492
609520 497309284 501000740 502716522 507298490 515725728 518394462 520058880 521436338 522338850 522482624 524354184 525034545 544389600 545335078 549702127
553607912 553878990 556225065 564656288 570128125 570280550 571450463 572132890 572923812 573525757 574294812 582646028 584717518 593336720 593564229 5940925
35 596473080 600177272 602749620 606529136 608854464 609964704 610656950 610663781 611170272 611438940 617200837 621741888 636044222 637937937 639990582 6406
24239 641916375 642180479 643673545 645027504 655306387 670192186 675775602 676051376 683195796 689763668 692870400 705332895 711785145 749812000 755923542 7
61240484 765264676 765399895 768384482 770605490 770623232 776672832 776862576 781020924 782058756 782156772 787616004 794268780 820689546 825084538 83764290
6 856195935 856952754 868248990 903287755 909654387 911580120 923342924 985600710 1067556196
C:\Bmb\Dominte_Mihai_Proiect-APD\Debug\Dominte_Mihai_Proiect-APD.exe (process 3380) exited with code 0.
Press any key to close this window . . .
Căutați
6°C
ENG
INTL
20:29
28.03.2023



Concluzie:

Algoritmul Bubble Sort are un timp de rulare O(n^2) lucru care poate fi observat si prin testele de mai sus unde timpul de rulare creste de aproximativ 4 ori la fiecare dublare a dimensiunii esantionului.