підготував старший викладач каф. ЕОМ Козак Назар Богданович

Домашнє завдання №12

Скласти програму (C/C++), яка дозволяє відсортувати вхідні величини методом злиття. Безпосередньо сортування методом злиття виконується при об'єднанні частин вхідних даних, які в свою чергу сортуються за допомогою швидкого сортування з стандартної бібліотеки мови С. Кількість частин для злиття та розмір вхідного масиву даних обрати відповідно до варіанту.

Вибір варіанту

(Nж + Nг + 1) % 30 + 1

де: Nж – порядковий номер студента в групі, а Nг – номер групи(1,2,3,4,5,6,7,8 або 9)

Варіанти завдання

Варіант	Розмір масиву вхідних даних	Кількість частин
1	8192	32
2	4096	32
3	2048	32
4	1024	32
5	512	32
6	256	32
7	128	32
8	64	32
9	8192	16
10	4096	16
11	2048	16
12	1024	16
13	512	16
14	256	16
15	128	16
16	64	16
17	8192	8
18	4096	8
19	2048	8
20	1024	8
21	512	8
22	256	8
23	128	8
24	64	8
25	8192	4
26	4096	4
27	2048	4
28	1024	4
29	512	4
30	256	4

Приклад коду

Лістинг

```
#include <stdio.h>
#include <stdlib.h>
#define BLOCK_COUNT 4
#define BLOCK_SIZE 8
#define DATA_SIZE (BLOCK_COUNT * BLOCK_SIZE)

int compareFunction(const void* a, const void* b){
	const int *arg1 = (const int *)a;
	const int *arg2 = (const int *)b;

	return *arg1 - *arg2;
}
void assimilator(void * outputData, void * data){
	int blockIndex, outputIndex, selectedBlockIndex, indexes[BLOCK_COUNT] = { 0 };

	for (outputIndex = 0; outputIndex < DATA_SIZE; ++outputIndex){
		selectedBlockIndex = -1;
		for (blockIndex = 0; blockIndex < BLOCK_COUNT; ++blockIndex){
			if (indexes[blockIndex] < BLOCK_SIZE && (selectedBlockIndex == -1 ||
((int*)data)[blockIndex * BLOCK_SIZE + indexes[blockIndex]] <
((int*)data)[selectedBlockIndex * BLOCK_SIZE + indexes[selectedBlockIndex]])){
				selectedBlockIndex = blockIndex;
			}
		}

		((int*)outputData)[outputIndex] = ((int*)data)[selectedBlockIndex *
BLOCK_SIZE + indexes[selectedBlockIndex]];
		++indexes[selectedBlockIndex];
	}
}
void sort(void * outputData, void * inputData){
	int blockIndex = 0;

	for (blockIndex = 0; blockIndex < BLOCK_COUNT; ++blockIndex){
		qsort((int*)inputData + blockIndex * BLOCK_SIZE, BLOCK_SIZE, sizeof(int),
compareFunction);
	}

	assimilator(outputData, inputData);
}
void printVector(void * data, int count/* 0 - full DATA_SIZE*/){
	int index = 0;
	for (index = 0; (!count || index < count) && index < DATA_SIZE; index++){
		printf("%d ", ((int *)data)[index]);
	}
	printf("\n");
}
int inputData[DATA_SIZE] = { 1, 20, 29, 28, 10, 26, 25, 24,
			     23, 22, 21, 30, 19, 18, 17, 16,
			     15, 14, 13, 12, 11, 27, 9, 8,
			     7, 6, 5, 4, 3, 2, 0, 31 };
int outputData[DATA_SIZE];
int main(void){
	sort(outputData, inputData);
	printVector(outputData, 0);

	getchar();
	return EXIT_SUCCESS;
}
```