Домашнє завдання №6

Скласти програму (C/C++), яка з застосуванням алгоритмічної стратегії «перебір з поверненням»(англ. *backtracking*) дозволяє виконати таке розміщення фігур ферзів на шахівниці, що жодна з них не ставить під удар іншу.

Вибір варіанту

(Nж + Nг + 1) % 3+ 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 шахівниці	кількість ферзів
1	9x9	9
2	10x10	10
3	11x11	11

Приклад коду

Лістинг

```
#include "stdio.h"
#ifdef __linux__
#elif defined(_WIN32)
#include "windows.h"
#else
#endif

#define DS (~((unsigned int)~0 >> 1))

#define N 8

#define MARKER_SIZE sizeof(unsigned int)

unsigned char markerPrepare(void * marker, void * globalContext){
	unsigned int positionIndex, (*positions)[2] = (unsigned int(*)[2])globalContext;

	for (positionIndex = 0; positions[positionIndex][1] /*&&
positions[positionIndex][0]*/; ++positionIndex);

	*(unsigned int*)marker = positionIndex + 1;

	return *(unsigned int*)marker > N;
}
unsigned char addMarker(void * data, void * marker, void * globalContext, void *
localContext){
	unsigned int(*positions)[2] = (unsigned int(*)[2])globalContext;
	unsigned int positionIndex, accept, iIndex = 0, jIndex = 0, iIndex_ = 0, jIndex_ =
0, iDelta_ = 0, jDelta_ = 0, (*table)[N] = (unsigned int(*)[N])data;

	unsigned int * startPosition = (unsigned int(*))localContext;

	iIndex = startPosition[1];
	jIndex = startPosition[0];
```

```
	for (; iIndex < N; ++iIndex, jIndex = 0){
		for (; jIndex < N; ++jIndex){
			if (!table[iIndex][jIndex]){
				accept = 1;

				for (positionIndex = 0; accept && positions[positionIndex][1];
++positionIndex){

					iIndex_ = positions[positionIndex][1] - 1;
					jIndex_ = positions[positionIndex][0] - 1;

					accept = iIndex != iIndex_ && jIndex != jIndex_;
					if (accept){
						iDelta_ = iIndex - iIndex_;
						if (iDelta_ & DS) iDelta_ -= iDelta_ << 1; // 2
* iDelta_;

						jDelta_ = jIndex - jIndex_;
						if (jDelta_ & DS) jDelta_ -= jDelta_ << 1; // 2
* jDelta_;

						accept = iDelta_ != jDelta_;
					}
				}

				if (accept){
					table[iIndex][jIndex] = *(unsigned int *)marker;

					for (positionIndex = 0; positions[positionIndex][1]
/*&& positions[0][positionIndex][0]*/; ++positionIndex);
					positions[positionIndex][1] = iIndex + 1;
					positions[positionIndex][0] = jIndex + 1;

					startPosition[1] = iIndex;// +1;
					startPosition[0] = jIndex + 1;

					return 1;
				}
			}
		}
	}

	return 0;
}

void removeMarker(void * data, void * globalContext){
	unsigned int(*positions)[2] = (unsigned int(*)[2])globalContext;
	unsigned int positionIndex, (*table)[N] = (unsigned int(*)[N])data;

	for (positionIndex = 0; positions[positionIndex][1] /**/ &&
positions[positionIndex][0]/**/; ++positionIndex);
		if (positionIndex){
			--positionIndex;
			table[positions[positionIndex][1] - 1][positions[positionIndex][0] -
1] = 0;
			positions[positionIndex][1] = 0;
			positions[positionIndex][0] = 0;
		}
}

void printTable(unsigned int table[N][N]){
	unsigned int iIndex = 0, jIndex = 0;

	printf("\r\n");
	for (; iIndex < N; ++iIndex){
		printf(" ");
```

```
		for (jIndex = 0; jIndex < N; ++jIndex){
			printf("%d ", table[iIndex][jIndex]);
		}
		printf("\r\n");
	}
	printf("\r\n");

}

void * bt(void * data, void * marker, void * globalContext, void * localContext){
	char newContext[1024] = { 0 };
	char newMarker_[MARKER_SIZE];
	void *prevCheep = NULL, *newMarker = (void*)newMarker_, *returnValue = (void*)1;

	if (markerPrepare(marker, globalContext)) return NULL;

	while (addMarker(data, marker, globalContext, (void*)newContext) && (returnValue =
bt(data, newMarker, globalContext, (void*)newContext))){
		removeMarker(data, globalContext);
	}

	if (!returnValue) return NULL;

	return (void*)(1);
}

unsigned int table[N][N] = { { 0 } };
char globalContext[N * MARKER_SIZE * 2] = { 0 };
char localContext[2 * N * MARKER_SIZE * 2] = { 0 };

int main(){
	unsigned int marker;

	void *returnValue = bt((void *)table, (marker = 1, &marker), (void
*)globalContext, (void *)localContext);

	printTable(table);

#ifdef __linux__
	(void)getchar();
#elif defined(_WIN32)
	system("pause");
#else
#endif
	return 0;
}
```