```cpp
#include <iostream>
#include <Windows.h>

struct ELEM
{
	int value;
	char letter;
	ELEM(int value, char letter) :value(value)
	{
		this->letter = letter;
	}
	void print()
	{
		std::cout << value << ' ' << letter << '\n';
	}
};

using ptrELEM = ELEM*;

struct ARRAY
{
private:
	size_t size;
	ptrELEM* array;
public:
	ARRAY(size_t size);
	~ARRAY();
	void print();
	size_t get_size() const // const в конце метода говорит о том, что метод
				// ничего не меняет в объекте, для которого вызывается,
				// так что его можно вызывать и для константного объекта
	{
		return size;
	}
	ptrELEM* get_array()
	{
		return array;
	}
	ptrELEM get_elem(size_t i) const
	{
		ptrELEM result = nullptr;
		if (i >= 0 && i < size)
			result = array[i];
		return result;
	}
};
```

```
ARRAY::ARRAY(size_t size) :size(size)
{
	srand(GetTickCount());
	array = new ptrELEM[size];
	for (size_t i = 0; i < size; ++i)
	{
		array[i] = new ELEM(rand() % 100, 'a' + rand() % 26);
	}
}

ARRAY::~ARRAY()
{
	for (size_t i = 0; i < size; ++i)
		delete array[i];
	delete[] array;
}

void ARRAY::print()
{
	for (size_t i = 0; i < size; ++i)
		array[i]->print();
}

int count_even(ARRAY &arr)
{
	int count = 0;
	ptrELEM* begin = arr.get_array();
	size_t size = arr.get_size();
	for (ptrELEM* ptr = begin; ptr < begin + size; ++ptr)
	{
		if ((*ptr)->value % 2 == 0)
			++count;
	}
	return count;
}

int count_odd(const ARRAY &arr)
{
	int count = 0;
	size_t size = arr.get_size();
	//ptrELEM ptr;
	for (size_t i = 0; i < size; ++i)
	{
		//ptr = arr.get_elem(i); // вариант используется, если не гарантируется
		// что i будет корректным (0 <= i < size)
		//if (ptr && ptr->value % 2 != 0)
		if (arr.get_elem(i)->value % 2 != 0)
			++count;
	}
	return count;
}
```

```cpp
int main()
{
	ARRAY arr(5);
	arr.print();

	std::cout << "---------------\n" << count_even(arr) << '\n';
	std::cout << "---------------\n" << count_odd(arr) << '\n';

	std::cin.get();
	return 0;
}
```