```
// Вариант 13.
// Main.h

//---------------------------------------------------------------------------

#ifndef _LAB7_MAIN_H_
#define _LAB7_MAIN_H_
//---------------------------------------------------------------------------
#include <Classes.hpp>
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
//---------------------------------------------------------------------------
class TForm1 : public TForm
{
__published:  // IDE-managed Components
	TMemo *FullHashTable;
	TButton *GenerateTable;
	TEdit *CuttingNumber;
	TMemo *HashTableLess;
	TMemo *HashTableMore;
	TButton *Cut;
	void __fastcall GenerateTableClick(TObject *Sender);
	void __fastcall CutClick(TObject *Sender);
private:	// User declarations
public:		// User declarations
	__fastcall TForm1(TComponent* Owner);
};
//---------------------------------------------------------------------------
extern PACKAGE TForm1 *Form1;
//---------------------------------------------------------------------------
#endif



// Main.cpp

//---------------------------------------------------------------------------
#pragma hdrstop
#include <vcl.h>
#include "Main.h"
#include "HashTable.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
Table* hashTable;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
	: TForm(Owner) {
	hashTable = NULL;
}
//---------------------------------------------------------------------------
// Функция заполняет таблицу случайными числами.
void __fastcall TForm1::GenerateTableClick(TObject *Sender) {
	FullHashTable->Clear();
	hashTable = new Table;
	randomize();
	for (int i = 0; i < 20; i++) {
		hashTable->Add(new Zap(random(100), random(100)));
	}
	UnicodeString* tableStrings = hashTable->ToPrint();
	for(int i = 0; i < 10; i++) {
		FullHashTable->Lines->Add(tableStrings[i]);
	}
}
```

```
//---------------------------------------------------------------------------
// Функция выводит 2 таблицы, образовавшиеся при разделении основной.
void __fastcall TForm1::CutClick(TObject *Sender) {
	HashTableLess->Clear();
	HashTableMore->Clear();
	int cutNum = 0;
	try {
		cutNum = CuttingNumber->Text.ToInt();
	}
	catch(Exception* e) {
		ShowMessage(e->Message);
		return;
	}
	Table* hashTableLess = new Table;
	Table* hashTableMore = new Table;
	if(hashTable != NULL) {
		hashTable->Cut(hashTableLess, hashTableMore, cutNum);
	}
	UnicodeString *TableStringsLess, *TableStringsMore;
	TableStringsLess = hashTableLess->ToPrint();
	TableStringsMore = hashTableMore->ToPrint();
	for(int i = 0; i < 10; i++) {
		HashTableLess->Lines->Add(TableStringsLess[i]);
		HashTableMore->Lines->Add(TableStringsMore[i]);
	}
}
//---------------------------------------------------------------------------
```

```
// HashTable.h
// Файл содержит класс хеш-таблицы.
#ifndef _LAB7_HASHTABLE_H_
#define _LAB7_HASHTABLE_H_

#include "vcl.h"
#include "Zap.h"

class Table {
private:
	Zap* hashTable;

public:
	Table();
	UnicodeString* ToPrint();
	void Cut(Table*&, Table*&, int);
	static int HashCode(int);
	void Add(Zap*);
	Zap* Find(int);

};

#endif
```

```
// HashTable.cpp

#pragma hdrstop
#include "HashTable.h"

Table::Table() {
	hashTable = new Zap[10];
}

int Table::HashCode(int key) {
	return key%10;
```

```
}
// Добавление данных в соответствующую ячейку таблицы.
void Table::Add(Zap* data) {
	int hCode = HashCode(data->Key());
	if(!hashTable[hCode].isFull) {
		hashTable[hCode] = *data;
		hashTable[hCode].isFull = true;
	}
	else {                                  // Если ячейка занята,
		Zap* ceil = &hashTable[hCode];      // данные добавляются в конец
		while(ceil->next != NULL) {         // однонаправленного списка.
			ceil = ceil->next;
		}
		ceil->next = data;
	}
}

// Поиск элемента по ключу.
Zap* Table::Find(int key) {
	int hCode = HashCode(key);
	if(!hashTable[hCode].isFull){
		return NULL;
	}
	else {
		Zap* ceil = hashTable + hCode;
		while (ceil->next != NULL) {
			ceil = ceil->next;
		}
		return ceil;
	}
}

// Копирование ключей таблицы в массив строк.
UnicodeString* Table::ToPrint() {
	UnicodeString* strings = new UnicodeString[10];
	for(int i = 0; i < 10; i++) {
		strings[i] = "";
		if(!hashTable[i].isFull) {
			continue;
		}
		else {
			Zap* ceil = hashTable + i;
			strings[i] += IntToStr(ceil->Key());
			while (ceil->next != NULL) {
				ceil = ceil->next;
				strings[i] += "   ";
				strings[i] += IntToStr(ceil->Key());
			}
		}
	}
	return strings;
}

// Деление таблицы на две: с ключами больше num и меньше.
void Table::Cut(Table* &hashTableLess, Table* &hashTableMore, int num) {
	for(int i = 0; i < 10; i++) {
		if(!hashTable[i].isFull) {
			continue;
		} else {
			Zap* ceil = hashTable + i;
			Zap* ceilCopy = new Zap(ceil->Key(), ceil->Info());
			ceilCopy->Key() < num ? hashTableLess->Add(ceilCopy) :
				hashTableMore->Add(ceilCopy);
			while (ceil->next != NULL) {
				ceil = ceil->next;
```

```
                        ceilCopy = new Zap(ceil->Key(), ceil->Info());
                        ceil->Key() < num ? hashTableLess->Add(ceilCopy) :
                                hashTableMore->Add(ceilCopy);
                }
        }
    }
}


#pragma package(smart_init)


// Zap.h
// Файл содержит класс, в котором описана ячейка таблицы.

#ifndef _LAB7_ZAP_H_
#define _LAB7_ZAP_H_

class Zap {
private:
    int key;
    int info;

public:
    Zap(int key, int info);
    Zap();
    int Key();
    int Info();
    Zap* next;
    bool isFull;
};

#endif



// Zap.cpp

#pragma hdrstop
#include "conio.h"
#include "Zap.h"

Zap::Zap() {
    isFull = false;
    next = NULL;
}

Zap::Zap(int _key, int _info) {
    key = _key;
    info = _info;
    next = NULL;
    isFull = true;
}

Zap::Key() {
    return key;
}

Zap::Info() {
    return info;
}



#pragma package(smart_init)
```