Лабораторная работа №3

По дисциплине «Информатика»

Тема: Простое наследование и полиморфизм

Выполнил:

студент группы [EMPTY]

[YOUR NAME]

Принял:

[TEACHER NAME]

**Цель работы**: приобрести навыки в использовании простого наследования и полиморфизма.

Пример работы позднего связывания

**Структура программы**

* deps.h - содержит макросы, что позволяет программе работать как на Windows так и на Unix-подобных платформах
* deps.cpp - содержит определения Windows-зависимых функций
* classes.h - содержит объявления базового и наследуемого классов
* classes.cpp - содержит определения базового и наследуемого классов.
* main.cpp - содержит интерфейс по работе с данными классами

**Ниже представлен код программы**

// deps.h

#ifndef \_\_DEPS\_H

#define \_\_DEPS\_H

# ifdef \_\_WIN32

# include <windows.h>

# include <conio.h>

# define KB\_ENTER 13

# else

# include <unistd.h>

# include <termios.h>

# include <sys/ioctl.h>

# define KB\_ENTER 10

int putch (int c);

int getch (void);

int kbhit (void);

# endif

#define KB\_ESCAPE 27

#endif

// deps.cpp

#include <cstdio>

#include "deps.h"

#ifndef \_\_WIN32

int putch (int c)

{

return putchar (c);

}

int kbhit(void) {

static const int STDIN = 0;

static bool initialized = false;

if (! initialized) {

// Use termios to turn off line buffering

termios term;

tcgetattr(STDIN, &term);

term.c\_lflag &= ~ICANON;

tcsetattr(STDIN, TCSANOW, &term);

setbuf(stdin, NULL);

initialized = true;

}

int bytesWaiting;

ioctl(STDIN, FIONREAD, &bytesWaiting);

return bytesWaiting;

}

int getch (void)

{

int ch;

struct termios oldtc;

struct termios newtc;

tcgetattr(STDIN\_FILENO, &oldtc);

newtc = oldtc;

newtc.c\_lflag &= ~(ICANON | ECHO);

tcsetattr(STDIN\_FILENO, TCSANOW, &newtc);

ch=getchar();

tcsetattr(STDIN\_FILENO, TCSANOW, &oldtc);

return ch;

}

#endif

// classes.h

#ifndef \_\_LAB3\_H

#define \_\_LAB3\_H

#define HEIGHT 5

#define WIDTH 5

class Table {

public:

int x, y;

int table[HEIGHT][WIDTH];

Table (void);

/\* There's nothing allocate memeory for. \*/

~Table (void);

virtual void add (int val);

void del (int val);

void show (void);

};

class SortedTable: public Table {

public:

SortedTable (void);

~SortedTable (void);

virtual void add (int val);

};

#endif

// classes.cpp

#include <iostream>

#include "classes.h"

using namespace std;

static void swap(int\* xp, int\* yp)

{

int temp = \*xp;

\*xp = \*yp;

\*yp = temp;

}

// Function to perform Selection Sort

static void selectionSort(int \*arr, int n)

{

int i, j, min\_idx;

// One by one move boundary of unsorted subarray

for (i = 0; i < n - 1; i++) {

// Find the minimum element in unsorted array

min\_idx = i;

for (j = i + 1; j < n; j++)

if (arr[j] < arr[min\_idx])

min\_idx = j;

// Swap the found minimum element

// with the first element

swap(&arr[min\_idx], &arr[i]);

}

}

Table::Table (void)

{

x = 0;

y = 0;

/\* Fill table with with min values to

indicate it's ready to use. \*/

for (int i = 0; i < HEIGHT; ++i) {

for (int j = 0; j < WIDTH; ++j) {

table[i][j] = 0;

}

}

}

Table::~Table (void) { }

void Table::add (int val)

{

if (x == WIDTH) {

if (y == HEIGHT - 1) {

cout << "Ошибка: таблица заполнена\n\n" << endl;

return ;

}

y++;

x = 0;

}

table[y][x++] = val; /\* fill table cell. \*/

cout << "Элемент добавлен в массив\n\n";

}

void Table::del (int val)

{

for (int i = 0; i <= y; ++i) {

for (int j = 0; j < WIDTH; ++j) {

if (table[i][j] == val) {

table[i][j] = 0;

cout << "Элемент удален из массива\n\n";

return ;

}

}

}

cout << "Ошибка: нет значения " << val << "\n" << endl;

}

void Table::show (void) {

cout << "Таблица\n";

for (int i = 0; i < HEIGHT; ++i) {

for (int j = 0; j < WIDTH; ++j) {

cout << table[i][j] << "\t";

}

cout << "\n";

}

cout << endl;

}

SortedTable::SortedTable (void) { }

SortedTable::~SortedTable (void) { }

void SortedTable::add (int val)

{

if (x == WIDTH) {

if (y == HEIGHT - 1) {

cout << "Ошибка: Таблица заполнена\n" << endl;

return ;

}

y++;

x = 0;

}

table[y][x++] = val; /\* fill table cell. \*/

cout << "Элемент добавлен в массив\n\n";

/\* sort table's row ONLY if it's full. \*/

if (x != WIDTH) {

return ;

}

selectionSort (table[y], HEIGHT);

}

// main.cpp

#include <iostream>

#include <climits>

#include <cstdio>

#include "classes.h"

#include "deps.h"

using namespace std;

/\*

Создать базовый класс «таблица» из N\*N полей, определив в нем

1) функции добавления элемента в таблицу,

2) удаление элемента из таблицы по значению,

3) вывод таблицы на экран.

Породить от него класс «упорядоченная

таблица», переопределив функцию добавления элементов.

\*/

static void menu1 (void)

{

cout << "1 - обычная таблица\n";

cout << "2 - сортированная таблица\n";

cout << "ESC - выход\n\n";

}

static void menu2 (void)

{

cout << "1 - добавить элемент\n";

cout << "2 - удалить элемент\n";

cout << "3 - вывод на экран\n\n";

}

static int getnum (void)

{

int idx;

int c;

int i = 0;

char buff[20];

while ((c = getch ()) && c != '\n') {

if (c >= '0' && c <= '9') {

buff[i++] = c;

putch (c);

}

}

buff[i] = 0;

putch ('\n');

idx = atoi (buff);

return idx;

}

int main (void)

{

Table otable; /\* ordinary table. \*/

SortedTable stable; /\* sorted table. \*/

Table \*table; //указатель нужен для реализации полиморфизма

int key1, key2;

int val; /\* where insert value to. \*/

setlocale(0, "RUS"); // обеспесиваем вывод в консольное окно символов кириллицы

menu1 ();

do {

key1 = getch();

switch(key1) {

case '1':

table=&otable;

break;

case '2':

table=&stable;

break;

}

if (key1 != KB\_ESCAPE) {//выбираем действие для выбранного типа таблицы

menu2 ();

do {

key2 = getch();

switch (key2) {

case '1':

cout << "Введите число (add): ";

val = getnum ();

table->add(val);

break;

case '2':

cout << "Введите число (del): ";

val = getnum ();

table->del(val);

break;

case '3':

table->show();

break;

}

} while (key2 != KB\_ESCAPE);

}

cout << "Выход из таблицы\n\n";

menu1 ();

} while (key1 != KB\_ESCAPE);

return 0;

}

Экранные формы с примерами работы программы

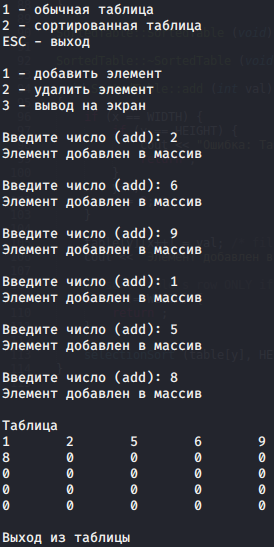
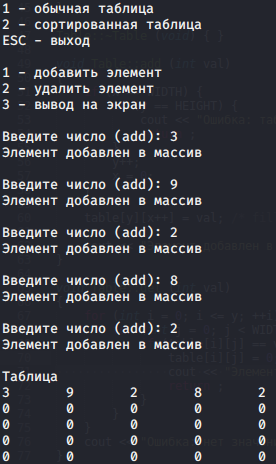


Рис. 1 Рис. 2

Рис. 1 отображает операции над обычной таблицей: добавление, удаление элементов и вывод. Сортировки в данном случае не происходит. Элементы отображаются в той же последовательности что были введены пользователем.

Рис. 2 отображает те же операции над сортированной таблицей. В данном случае после заполнения ряда, таблица делает сортировку от меньшего к большему.

Обработка ошибок

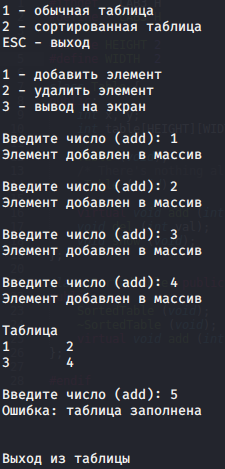
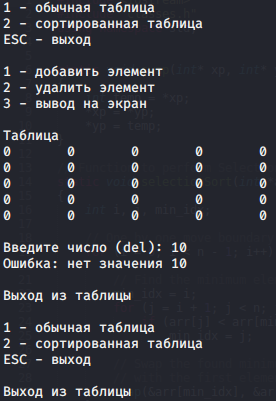


Рис . 3 Рис. 4

На Рис. 3 показано, как таблица обрабатывает ошибку связанную с удалением несуществующего элемента. В данном случае в таблице ничего не меняется. После выводится соответствующее сообщение пользователю.

На Рис. 4 показана обработка другой ошибки: переполнение. Если таблица полностью заполнена, при попытке добавить еще элемент программа выдает предупреждение пользователю. Данный механизм позволяет защитить данные в таблице.

Заключение

В данной лабораторной работе я приобрел навыки в использовании простого наследования и полиморфизма. И реализовал класс из моего варианта, что продемонстрировал механизм позднего связывания.