**МИНОБРНАУКИ РОССИИ**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ  
**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ  
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.Г.ШУХОВА»  
(БГТУ им. В.Г.Шухова)**

Кафедра программного обеспечения вычислительной техники и автоматизированных систем

Лабораторная работа №2

Дисциплина: Обьектно-ориентированное программирование

по теме «Модульное программирование. Интерфейсы.»

Выполнил: ст. группы ПВ-21  
Ковалев Павел

Проверил: Буханов Д.Г.

Белгород 2020

**Цель работы**: Получение навыков модульной декомпозиции предметной области, создания модулей. Разработка интерфейсов.

**Задание:** Разработать программу, состоящую из трех модулей в соответствии с указанным вариантом задания. Первый модуль – основной код программы; второй содержит интерфейсы; третий модуль – реализацию этих интерфейсов. Количество структур данных ("объектов") не менее пяти.

Вариант 8

Программа «Файловый чат»

**Файл chat.hpp**

#include <stdio.h>

#include <string.h>

#include <sys/types.h>

#include <sys/stat.h>

#include <unistd.h>

#include <errno.h>

#include <fcntl.h>

#include <stdlib.h>

#include <time.h>

// Получение системного времени в формате,

// заданном formatstring

struct time\_getter{

char \*formatstring;

int get\_time(char \*, int);

void set\_formatting(const char \*);

};

// Получение переданных опций

// Поддерживаются две опции

// -f имя\_файла -u имя\_пользователя

struct opts{

char \*filename;

char \*username;

int get\_opts(int argc, char \*\*argv);

};

// Работа с файлом - открытие/закрытие,

// отметка последней записи, запись и чтение,

// проверка изменений

struct file\_descr{

char \*name;

FILE \*stream;

struct timespec last\_modified;

long last\_pos = 0;

void set\_name(const char \*);

int open\_file();

void close\_file();

int has\_updates();

void read\_new\_lines(char \*, int);

void write\_lines(const char \*);

};

// Форматирование вывода в виде

// <время> [имя\_пользователя] сообщение

struct format{

// Время в формате час:минута (24ч)

struct time\_getter time{.formatstring = "%R"};

// Ширина вывода

const int max\_length = 80;

// Цвета пользователя и собеседника

const char usr\_color[8] = "\033[1;31m";

const char chat\_color[8] = "\033[1;33m";

const char reset\_color[5] = "\033[0m";

void usr\_out (const char \*, const char \*, char \*, int);

void plain\_out(const char \*, const char \*, char \*, int);

};

// Опрос файла и консоли, вывод

struct main\_loop{

// Интервал опроса ввода пользователя (1 мкс)

const timespec sleep\_interval{.tv\_sec = 0, .tv\_nsec = 1000};

// Буфер пользовательского ввода

static const int buf\_size = 512;

// Необработанный ввод, вывод для файла и для пользователя

char raw\_buf[buf\_size];

char out\_buf[buf\_size];

char usr\_buf[buf\_size];

void loop(char \*, struct file\_descr \*);

};

**Файл chat.cpp**

#include "chat.hpp"

int time\_getter::get\_time(char \*buffer, int max\_size){

time\_t rawtime;

struct tm \*timeinfo;

time(&rawtime);

timeinfo = localtime(&rawtime);

return strftime(buffer, max\_size, formatstring, timeinfo);

}

void time\_getter::set\_formatting(const char \*fmtstring){

int length = strlen(fmtstring);

formatstring = new char[length + 1];

strcpy(formatstring, fmtstring);

}

int opts::get\_opts(int argc, char \*\*argv){

filename = username = NULL;

int opt;

while ((opt = getopt(argc, argv, "f:u:")) != -1){

switch(opt){

case 'f':

filename = optarg;

break;

case 'u':

username = optarg;

break;

case '?':

if (optopt == 'i' || optopt == 'o'){

fprintf(stderr, "Опция -%c ожидает аргумент. \n", optopt);

}else{

fprintf(stderr, "Неизвестная опция -%c. \n", optopt);

}

return 0;

default:

abort();

}

}

return !filename || !username;

}

void file\_descr::set\_name(const char \*fname){

int len = strlen(fname);

name = new char[len+1];

strcpy(name, fname);

}

int file\_descr::open\_file(){

struct stat st;

stream = fopen(name, "a+");

if(stream == NULL) return 0;

if (stat(name, &st) == -1){

return 0;

}

last\_modified = st.st\_mtim;

return 1;

}

void file\_descr::close\_file(){

fclose(stream);

struct stat st;

stat(name, &st);

last\_modified = st.st\_mtim;

}

void file\_descr::read\_new\_lines(char \*buffer, int buffer\_size){

fseek(stream, last\_pos, SEEK\_SET);

int c, i;

for (i = 0; (c = fgetc(stream)) != EOF && i < buffer\_size; i++){

buffer[i] = c;

}

buffer[i] = '\0';

last\_pos = ftell(stream);

}

void file\_descr::write\_lines(const char \*buffer){

for (int i = 0; buffer[i]; i++){

fputc(buffer[i], stream);

}

last\_pos = ftell(stream);

}

int file\_descr::has\_updates(){

struct stat st;

if (stat(name, &st) == -1){

return 0;

}

if (st.st\_mtim.tv\_sec != last\_modified.tv\_sec ||

st.st\_mtim.tv\_nsec != last\_modified.tv\_nsec){

last\_modified = st.st\_mtim;

return 1;

}else{

return 0;

}

}

void format::plain\_out(const char \*username, const char \*str, char \*buffer, int buffer\_size){

const int tbuf\_size = 10;

char time\_buf[tbuf\_size];

time.get\_time(time\_buf, tbuf\_size);

snprintf(buffer, buffer\_size, " <%s> [%s] %s", time\_buf, username, str);

}

void format::usr\_out(const char \*color, const char \*str, char \*buffer, int buffer\_size){

int len\_left = max\_length;

int buff\_pos = 0;

for (int i = 0; color[i] && buff\_pos < buffer\_size; i++){

buffer[buff\_pos++] = color[i];

}

for (int i = 0; str[i] && buff\_pos < buffer\_size; i++){

buffer[buff\_pos++] = str[i];

len\_left--;

if (!len\_left){

buffer[buff\_pos++] = '\n';

len\_left = max\_length;

}

}

for (int i = 0; reset\_color[i] && buff\_pos < buffer\_size; i++){

buffer[buff\_pos++] = reset\_color[i];

}

buffer[buff\_pos] = '\0';

}

void main\_loop::loop(char \*username, struct file\_descr \*file){

struct format formatter;

// fd - дескриптор стандартного ввода

int fd = fileno(stdin);

// Добавляем к флагам дескриптора ввода флаг неблокирующего ввода

const int flags = fcntl(fd, F\_GETFL, 0);

fcntl(fd, F\_SETFL, flags | O\_NONBLOCK);

// Цикл реализует неблокирующее чтение

// пользовательского ввода и обновление

// файла с чатом

ssize\_t nbytes = 0;

for (;;){

if (file->has\_updates()){

file->open\_file();

file->read\_new\_lines(raw\_buf, buf\_size);

file->close\_file();

formatter.usr\_out(formatter.chat\_color, raw\_buf, usr\_buf, buf\_size);

puts(usr\_buf);

}

if ((nbytes = read(fd, raw\_buf, sizeof(raw\_buf))) < 0){

if (errno != EWOULDBLOCK){

// Ошибка ввода

fprintf(stderr, "Input error!\n");

exit(1);

}

}else{

raw\_buf[nbytes] = '\0';

if (!strcmp(raw\_buf, "exit")){

printf("Exiting...\n");

exit(0);

}

// Произошло событие ввода

formatter.plain\_out(username, raw\_buf, out\_buf, buf\_size);

formatter.usr\_out(formatter.usr\_color, out\_buf, usr\_buf, buf\_size);

file->open\_file();

file->write\_lines(out\_buf);

file->close\_file();

puts(usr\_buf);

}

nanosleep(&sleep\_interval, NULL);

}

}

**Файл task.cpp**

#include "chat.hpp"

int main(int argc, char \*\*argv){

struct file\_descr file;

char \*username, \*filename;

struct opts options;

if (options.get\_opts(argc, argv)){

printf("Недостаточно аргументов\n");

exit(1);

}

file.set\_name(options.filename);

struct main\_loop mainloop;

mainloop.loop(options.username, &file);

return 0;

}