**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»

1. **Файл chat.hpp**
2. #include <stdio.h>
3. #include <string.h>
4. #include <sys/types.h>
5. #include <sys/stat.h>
6. #include <unistd.h>
7. #include <errno.h>
8. #include <fcntl.h>
9. #include <stdlib.h>
10. #include <time.h>
11. // Получение системного времени в формате,
12. // заданном formatstring
13. struct time\_getter{
14. char \*formatstring;
15. int get\_time(char \*, int);
16. void set\_formatting(const char \*);
17. };
18. // Получение переданных опций
19. // Поддерживаются две опции
20. // -f имя\_файла -u имя\_пользователя
21. struct opts{
22. char \*filename;
23. char \*username;
24. int get\_opts(int argc, char \*\*argv);
25. };
26. // Работа с файлом - открытие/закрытие,
27. // отметка последней записи, запись и чтение,
28. // проверка изменений
29. struct file\_descr{
30. char \*name;
31. FILE \*stream;
32. struct timespec last\_modified;
33. long last\_pos = 0;
34. void set\_name(const char \*);
35. int open\_file();
36. void close\_file();
37. int has\_updates();
38. void read\_new\_lines(char \*, int);
39. void write\_lines(const char \*);
40. };
41. // Форматирование вывода в виде
42. // <время> [имя\_пользователя] сообщение
43. struct format{
44. // Время в формате час:минута (24ч)
45. struct time\_getter time{.formatstring = "%R"};
46. // Ширина вывода
47. const int max\_length = 80;
48. // Цвета пользователя и собеседника
49. const char usr\_color[8] = "\033[1;31m";
50. const char chat\_color[8] = "\033[1;33m";
51. const char reset\_color[5] = "\033[0m";
52. void usr\_out (const char \*, const char \*, char \*, int);
53. void plain\_out(const char \*, const char \*, char \*, int);
54. };
55. // Опрос файла и консоли, вывод
56. struct main\_loop{
57. // Интервал опроса ввода пользователя (1 мкс)
58. const timespec sleep\_interval{.tv\_sec = 0, .tv\_nsec = 1000};
59. // Буфер пользовательского ввода
60. static const int buf\_size = 512;
61. // Необработанный ввод, вывод для файла и для пользователя
62. char raw\_buf[buf\_size];
63. char out\_buf[buf\_size];
64. char usr\_buf[buf\_size];
65. void loop(char \*, struct file\_descr \*);
66. };
67. **Файл chat.cpp**
68. #include "chat.hpp"
69. int time\_getter::get\_time(char \*buffer, int max\_size){
70. time\_t rawtime;
71. struct tm \*timeinfo;
72. time(&rawtime);
73. timeinfo = localtime(&rawtime);
74. return strftime(buffer, max\_size, formatstring, timeinfo);
75. }
76. void time\_getter::set\_formatting(const char \*fmtstring){
77. int length = strlen(fmtstring);
78. formatstring = new char[length + 1];
79. strcpy(formatstring, fmtstring);
80. }
81. int opts::get\_opts(int argc, char \*\*argv){
82. filename = username = NULL;
83. int opt;
84. while ((opt = getopt(argc, argv, "f:u:")) != -1){
    * 1. switch(opt){
         1. case 'f':
            1. filename = optarg;
            2. break;
         2. case 'u':
            1. username = optarg;
            2. break;
         3. case '?':
            1. if (optopt == 'i' || optopt == 'o'){

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

* + - * 1. }else{

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

* + - * 1. }
        2. return 0;
      1. default:
         1. abort();
    1. }

1. }
2. return !filename || !username;
3. }
4. void file\_descr::set\_name(const char \*fname){
5. int len = strlen(fname);
6. name = new char[len+1];
7. strcpy(name, fname);
8. }
9. int file\_descr::open\_file(){
10. struct stat st;
11. stream = fopen(name, "a+");
12. if(stream == NULL) return 0;
13. if (stat(name, &st) == -1){
    * 1. return 0;
14. }
15. last\_modified = st.st\_mtim;
16. return 1;
17. }
18. void file\_descr::close\_file(){
19. fclose(stream);
20. struct stat st;
21. stat(name, &st);
22. last\_modified = st.st\_mtim;
23. }
24. void file\_descr::read\_new\_lines(char \*buffer, int buffer\_size){
25. fseek(stream, last\_pos, SEEK\_SET);
26. int c, i;
27. for (i = 0; (c = fgetc(stream)) != EOF && i < buffer\_size; i++){
    * 1. buffer[i] = c;
28. }
29. buffer[i] = '\0';
30. last\_pos = ftell(stream);
31. }
32. void file\_descr::write\_lines(const char \*buffer){
33. for (int i = 0; buffer[i]; i++){
    * 1. fputc(buffer[i], stream);
34. }
35. last\_pos = ftell(stream);
36. }
37. int file\_descr::has\_updates(){
38. struct stat st;
39. if (stat(name, &st) == -1){
    * 1. return 0;
40. }
41. if (st.st\_mtim.tv\_sec != last\_modified.tv\_sec ||
    1. st.st\_mtim.tv\_nsec != last\_modified.tv\_nsec){
       1. last\_modified = st.st\_mtim;
       2. return 1;
42. }else{
    * 1. return 0;
43. }
44. }
45. void format::plain\_out(const char \*username, const char \*str, char \*buffer, int buffer\_size){
46. const int tbuf\_size = 10;
47. char time\_buf[tbuf\_size];
48. time.get\_time(time\_buf, tbuf\_size);
49. snprintf(buffer, buffer\_size, " <%s> [%s] %s", time\_buf, username, str);
50. }
51. void format::usr\_out(const char \*color, const char \*str, char \*buffer, int buffer\_size){
52. int len\_left = max\_length;
53. int buff\_pos = 0;
54. for (int i = 0; color[i] && buff\_pos < buffer\_size; i++){
    * 1. buffer[buff\_pos++] = color[i];
55. }
56. for (int i = 0; str[i] && buff\_pos < buffer\_size; i++){
    * 1. buffer[buff\_pos++] = str[i];
      2. len\_left--;
      3. if (!len\_left){
         1. buffer[buff\_pos++] = '\n';
         2. len\_left = max\_length;
      4. }
57. }
58. for (int i = 0; reset\_color[i] && buff\_pos < buffer\_size; i++){
    * 1. buffer[buff\_pos++] = reset\_color[i];
59. }
60. buffer[buff\_pos] = '\0';
61. }
62. void main\_loop::loop(char \*username, struct file\_descr \*file){
63. struct format formatter;
64. // fd - дескриптор стандартного ввода
65. int fd = fileno(stdin);
66. // Добавляем к флагам дескриптора ввода флаг неблокирующего ввода
67. const int flags = fcntl(fd, F\_GETFL, 0);
68. fcntl(fd, F\_SETFL, flags | O\_NONBLOCK);
69. // Цикл реализует неблокирующее чтение
70. // пользовательского ввода и обновление
71. // файла с чатом
72. ssize\_t nbytes = 0;
73. for (;;){
    * 1. if (file->has\_updates()){
         1. file->open\_file();
         2. file->read\_new\_lines(raw\_buf, buf\_size);
         3. file->close\_file();
         4. formatter.usr\_out(formatter.chat\_color, raw\_buf, usr\_buf, buf\_size);
         5. puts(usr\_buf);
      2. }
      3. if ((nbytes = read(fd, raw\_buf, sizeof(raw\_buf))) < 0){
         1. if (errno != EWOULDBLOCK){
      4. // Ошибка ввода
         + 1. fprintf(stderr, "Input error!\n");
           2. exit(1);
    1. }
74. }else{
    1. raw\_buf[nbytes] = '\0';
    2. if (!strcmp(raw\_buf, "exit")){
       1. printf("Exiting...\n");
       2. exit(0);
    3. }
       * 1. // Произошло событие ввода
       1. formatter.plain\_out(username, raw\_buf, out\_buf, buf\_size);
       2. formatter.usr\_out(formatter.usr\_color, out\_buf, usr\_buf, buf\_size);
    4. file->open\_file();
       * 1. file->write\_lines(out\_buf);
       1. file->close\_file();
       2. puts(usr\_buf);
75. }
    * 1. nanosleep(&sleep\_interval, NULL);
76. }
77. }
78. **Файл task.cpp**
79. #include "chat.hpp"
80. int main(int argc, char \*\*argv){
81. struct file\_descr file;
82. char \*username, \*filename;
83. struct opts options;
84. if (options.get\_opts(argc, argv)){
85. printf("Недостаточно аргументов\n");
86. exit(1);
87. }
88. file.set\_name(options.filename);
89. struct main\_loop mainloop;
90. mainloop.loop(options.username, &file);
91. return 0;
92. }