Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение

высшего образования

«**Пермский национальный исследовательский**

**политехнический университет»**

Кафедра «Информационные технологии и автоматизированные

системы»

**ОТЧЁТ**

**по лабораторной работе №18.8**

Дисциплина: «Информатика»

Тема: “Объектно-ориентированное программирование.

Программа, управляемая событиями.”

Вариант 13

Выполнил:

Студент группы ИВТ-20-2б

Сафронов Владислав Владиславович

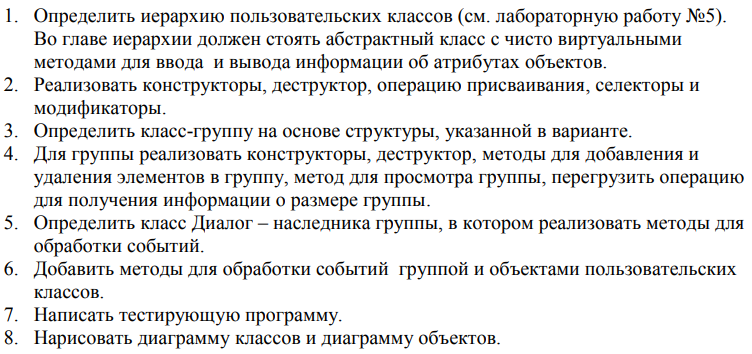
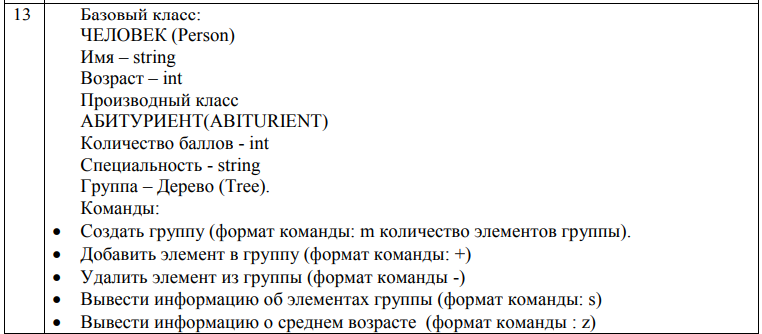
Проверила:

Доцент кафедры ИТАС

Полякова О.А.

**Пермь, 2021**

**Постановка задачи**

****

**Анализ задачи**

1. Для решения задачи необходимо:
   1. Организовать абстрактный класс Object с чисто виртуальными функциями для вывода данных на консоль и ввода данных.
   2. Организовать класс List – контейнер для объектов других классов.
   3. Организовать класс TEvent для работы с событиями.
   4. Организовать класс Dialog для работы с событиями.
   5. Организовать класс Person с полями для ввода и хранения ФИО (name) и возраста (age).
   6. Организовать класс Abiturient с полями для ввода и хранения специальности (speciality) и баллов ЕГЭ (exams).
2. В ходе работы были использованы следующие типы данных:
   1. Экземпляр класса Dialog – D, над которм будут производиться действия.
3. Dialog D;
   1. В функции GetEvent() класса Dialog используется switch() для переключения команды для метода HandleEvent()
4. void Dialog::GetEvent(TEvent& event)
5. {
6. string OpInt = "+-/mszq";
7. string s;
8. cout << ">";
9. getline(cin, s);
11. if (OpInt.find(s[0]) != -1)
12. {
13. event.what = evMessage;
14. switch (s[0])
15. {
16. case 'm': event.command = cmMake; break;
17. case '+': event.command = cmAdd; break;
18. case '-': event.command = cmDel; break;
19. case 's': event.command = cmShow; break;
20. case '/': event.command = cmName; break;
21. case 'z': event.command = cmAge; break;
22. case 'q': event.command = cmQuit; break;
23. }
24. if (s.length() > 1)
25. {
26. string param = s.substr(1, s.length() - 1);
27. event.a = atoi(param.c\_str());
28. }
29. }
30. else event.what = evNothing;
31. }
    1. В функции HandleEvent() класса Dialog используется switch() для обработки полученного события
32. void Dialog::HandleEvent(TEvent& event)
33. {
34. if (event.what == evMessage)
35. {
36. switch (event.command)
37. {
38. case cmMake: {
39. size = event.a;
40. beg = new Object \* [size];
41. cur = 0;
42. ClearEvent(event);
43. break;
44. }
45. case cmAdd: {
46. Add();
47. ClearEvent(event);
48. break;
49. }
50. case cmDel: {
51. Del();
52. ClearEvent(event);
53. break;
54. }
55. case cmShow: {
56. Show();
57. ClearEvent(event);
58. break;
59. }
60. case cmAge: {
61. ShowMidAge();
62. break;
63. }
64. case cmQuit: {
65. EndState = 1;
66. ClearEvent(event);
67. break;
68. }
69. default: List::HandleEvent(event);
70. }
71. }
72. }
73. Для решения задачи данные были представлены в следующем виде:
    1. Работа с событиями происходит через структуру TEvent с полем what, отвечающим за тип события. Вторая часть задаёт информацию, передаваемую с событием.
74. struct TEvent
75. {
76. int what;
77. union
78. {
79. int command;
80. struct
81. {
82. int message;
83. int a;
84. };
85. };
86. };
87. Для операций ввода и вывода использовались следующие операторы и функции:
    1. Для выбора события вводится ввод осуществляется через функцию getline() в переменную s. Пользователю предлагается выбор из операций создания группы, удаления элемента из группы, добавления элемента в группу, вывода элементов группы и выхода из программы.
88. void Dialog::GetEvent(TEvent& event)
89. {
90. string OpInt = "+-/mszq";
91. string s;
92. cout << ">";
93. getline(cin, s);
95. if (OpInt.find(s[0]) != -1)
96. {
97. event.what = evMessage;
98. switch (s[0])
99. {
100. case 'm': event.command = cmMake; break;
101. case '+': event.command = cmAdd; break;
102. case '-': event.command = cmDel; break;
103. case 's': event.command = cmShow; break;
104. case '/': event.command = cmName; break;
105. case 'z': event.command = cmAge; break;
106. case 'q': event.command = cmQuit; break;
107. }
108. if (s.length() > 1)
109. {
110. string param = s.substr(1, s.length() - 1);
111. event.a = atoi(param.c\_str());
112. }
113. }
114. else event.what = evNothing;
115. }
     1. Для ввода экземпляров классов Person и Abiturient используется метод Add класса List.
116. void List::Add()
117. {
118. if (size == 0)
119. {
120. cout << "Please make a List First\n";
121. return;
122. }
123. if (size == cur)
124. {
125. cout << "List is full\n";
126. return;
127. }
128. Object\* p;
130. cout << "1. Person" << endl;
131. cout << "2. Abiturient" << endl;
132. int y;
133. cout << "Choice: "; cin >> y;
134. if (y == 1)
135. {
136. Person\* a = new Person;
137. a->Input();
138. p = a;
140. if (cur < size)
141. {
142. beg[cur] = p;
143. cur++;
144. }
145. }
146. else if (y == 2)
147. {
148. Abiturient\* b = new Abiturient;
149. b->Input();
150. p = b;
151. if (cur < size)
152. {
153. beg[cur] = p;
154. cur++;
155. }
156. }
157. else return;
158. }
     1. Для вывода на консоль введенных объектов необходимо при выборе операции ввести “s”, тогда запустится метод Show() для класса List.
159. void List::Show()
160. {
161. if (cur == 0) cout << "Empty" << endl;
162. else
163. {
164. Object\*\* p = beg;
165. for (int i = 0; i < cur; i++)
166. {
167. (\*p)->Show();
168. p++;
169. }
170. }
172. }
173. Поставленные задачи будут решены следующими действиями:
     1. В функции main() вызывается метод Execute() для объекта D.
174. В данном методе создаётся объект event структуры TEvent – событие, с которым будут выполняться действия. В цикле do-while() вызываются методы GetEvent(), HandleEvenet() и ClearEvent().
175. void Dialog::Execute()
176. {
177. TEvent event;
178. do {
179. GetEvent(event);
180. HandleEvent(event);
181. if (event.what != evNothing) ClearEvent(event);
182. } while (EndState == 0);
183. }
     1. Метод GetEvent() получает событие: пользователь может создать группу удалить элемент из группы, добавить элемент в группу, вывести элементы группы, выйти из программы. Выбор осуществляется через switch(). Однако, чтобы попасть в ветвление switch() происходить проверка введённых данных в if(): если из заданной программистом строчки из операций находится символ введённый пользователем через метод find(), то происходит проход в ветвление, иначе полю what присваивается значение пустого события.

В случае ввода создания группы, нужно ввести дополнительный параметр через разделитель – размер группы (количество элементов в контейнере) через метод класса string – substr(), далее эта подстрока преобразуется в число через функцию atoi и данное значение присваивается параметру a класса TEvent.

1. void Dialog::GetEvent(TEvent& event)
2. {
3. string OpInt = "+-/mszq";
4. string s;
5. cout << ">";
6. getline(cin, s);
8. if (OpInt.find(s[0]) != -1)
9. {
10. event.what = evMessage;
11. switch (s[0])
12. {
13. case 'm': event.command = cmMake; break;
14. case '+': event.command = cmAdd; break;
15. case '-': event.command = cmDel; break;
16. case 's': event.command = cmShow; break;
17. case '/': event.command = cmName; break;
18. case 'z': event.command = cmAge; break;
19. case 'q': event.command = cmQuit; break;
20. }
21. if (s.length() > 1)
22. {
23. string param = s.substr(1, s.length() - 1);
24. event.a = atoi(param.c\_str());
25. }
26. }
27. else event.what = evNothing;
28. }
    1. В функции HandleEvent() происходит обработка полученного события из метода GetEvent(). Происходит провека: если в поле what записано не пустое событие, то происходит заход в по ветке true к ветвлению switch()
29. void Dialog::HandleEvent(TEvent& event)
30. {
31. if (event.what == evMessage)
32. {
33. switch (event.command) {…}
34. }
35. }

В ветвлении switch() в зависимости от выбора пользователя вызываются методы для обработки событий: если пользователь ввёл “m” – создание группы, создаётся динамический массив класса Object, наследниками которого являются классы Print, Book; если пользователь ввёл “+”, то вызывается метод push() для добавления элемента; если пользователь ввёл “-“, происходит вызов метода pop(), который удаляет элемент; если пользователь ввёл “q”, вызывается метод EndExec(), в котором параметру EndState присваивается 1 и происходит выход из цикла do-while() метода Execute().

1. switch (event.command)
2. {
3. case cmMake: {
4. size = event.a;
5. beg = new Object \* [size];
6. cur = 0;
7. ClearEvent(event);
8. break;
9. }
10. case cmAdd: {
11. Add();
12. ClearEvent(event);
13. break;
14. }
15. case cmDel: {
16. Del();
17. ClearEvent(event);
18. break;
19. }
20. case cmShow: {
21. Show();
22. ClearEvent(event);
23. break;
24. }
25. case cmAge: {
26. ShowMidAge();
27. break;
28. }
29. case cmQuit: {
30. EndState = 1;
31. ClearEvent(event);
32. break;
33. }
34. default: List::HandleEvent(event);
35. }
    1. Метод ClearEvent() присваивает полю what константу о пустом событие – evNothing.
36. void Dialog::ClearEvent(TEvent& event)
37. {
38. event.what = evNothing;
39. }

**Блок-схема программы**

**Код программы**

**Файл Abiturient.h:**

1. #pragma once
2. #include "Person.h"
4. class Abiturient :
5. public Person
6. {
7. protected:
8. string speciality;
9. int exams;
10. public:
11. Abiturient(void);
12. Abiturient(string, int, string, int);
13. Abiturient(const Abiturient&);
14. ~Abiturient(void);
15. void Show();
16. void Input();
17. string Get\_Speciality() { return speciality; }
18. int Get\_Exams() { return exams; }
19. void Set\_Speciality(string);
20. void Set\_Exams(int);
21. Abiturient& operator=(const Abiturient&);
22. };

**Файл Abiturient.cpp:**

1. #include "Abiturient.h"
3. Abiturient::Abiturient(void)
4. {
5. speciality = "";
6. exams = 0;
7. }
8. Abiturient::Abiturient(string n, int a, string s, int e) :Person(n, a)
9. {
10. speciality = s;
11. exams = e;
12. }
13. Abiturient::Abiturient(const Abiturient& a)
14. {
15. name = a.name;
16. age = a.age;
17. speciality = a.speciality;
18. exams = a.exams;
19. }
20. Abiturient::~Abiturient(void)
21. {}
22. void Abiturient::Set\_Speciality(string s)
23. {
24. speciality = s;
25. }
26. void Abiturient::Set\_Exams(int e)
27. {
28. exams = e;
29. }
30. Abiturient& Abiturient::operator=(const Abiturient& a)
31. {
32. if (&a == this) return \*this;
33. name = a.name;
34. age = a.age;
35. speciality = a.speciality;
36. exams = a.exams;
37. return \*this;
38. }
39. void Abiturient::Show()
40. {
41. cout << "\nName: " << name;
42. cout << "\nAge: " << age;
43. cout << "\nSpeciality: " << speciality;
44. cout << "\nExams: " << exams << endl;
45. }
46. void Abiturient::Input()
47. {
48. cout << "\nEnter name: "; cin.ignore(); getline(cin, name);
49. cout << "Enter age: "; cin >> age;
50. cout << "Enter speciality: "; cin.ignore(); getline(cin, speciality);
51. cout << "Enter exams: "; cin >> exams;
52. cin.get();
53. }

**Файл Dialog.h:**

1. #pragma once
2. #include "List.h"
3. #include "Event.h"
4. class Dialog:
5. public List
6. {
7. protected:
8. int EndState;
9. public:
10. Dialog(void);
11. ~Dialog(void);
12. void GetEvent(TEvent& event);
13. void Execute();
14. void HandleEvent(TEvent& event);
15. void ClearEvent(TEvent& event);
16. };

**Файл Dialog.cpp:**

1. #include "Dialog.h"
2. #include "Event.h"
3. #include <iostream>
4. #include <string>
5. using namespace std;
7. Dialog::Dialog(void):List()
8. {
9. EndState = 0;
10. }
11. Dialog::~Dialog(void)
12. {}
13. void Dialog::GetEvent(TEvent& event)
14. {
15. string OpInt = "+-/mszq";
16. string s;
17. cout << ">";
18. getline(cin, s);
20. if (OpInt.find(s[0]) != -1)
21. {
22. event.what = evMessage;
23. switch (s[0])
24. {
25. case 'm': event.command = cmMake; break;
26. case '+': event.command = cmAdd; break;
27. case '-': event.command = cmDel; break;
28. case 's': event.command = cmShow; break;
29. case '/': event.command = cmName; break;
30. case 'z': event.command = cmAge; break;
31. case 'q': event.command = cmQuit; break;
32. }
33. if (s.length() > 1)
34. {
35. string param = s.substr(1, s.length() - 1);
36. event.a = atoi(param.c\_str());
37. }
38. }
39. else event.what = evNothing;
40. }
42. void Dialog::Execute()
43. {
44. TEvent event;
45. do {
46. GetEvent(event);
47. HandleEvent(event);
48. if (event.what != evNothing) ClearEvent(event);
49. } while (EndState == 0);
50. }
52. void Dialog::ClearEvent(TEvent& event)
53. {
54. event.what = evNothing;
55. }
57. void Dialog::HandleEvent(TEvent& event)
58. {
59. if (event.what == evMessage)
60. {
61. switch (event.command)
62. {
63. case cmMake: {
64. size = event.a;
65. beg = new Object \* [size];
66. cur = 0;
67. ClearEvent(event);
68. break;
69. }
70. case cmAdd: {
71. Add();
72. ClearEvent(event);
73. break;
74. }
75. case cmDel: {
76. Del();
77. ClearEvent(event);
78. break;
79. }
80. case cmShow: {
81. Show();
82. ClearEvent(event);
83. break;
84. }
85. case cmAge: {
86. ShowMidAge();
87. break;
88. }
89. case cmQuit: {
90. EndState = 1;
91. ClearEvent(event);
92. break;
93. }
94. default: List::HandleEvent(event);
95. }
96. }
97. }

**Файл Event.h:**

1. #pragma once
3. const int evNothing = 0;
4. const int evMessage = 100;
5. const int cmAdd = 1;
6. const int cmDel = 2;
7. const int cmGet = 3;
8. const int cmShow = 4;
9. const int cmAge = 5;
10. const int cmMake = 6;
11. const int cmName = 7;
12. const int cmQuit = 101;
14. struct TEvent
15. {
16. int what;
17. union
18. {
19. int command;
20. struct
21. {
22. int message;
23. int a;
24. };
25. };
26. };

**Файл Object.h:**

1. #pragma once
2. #include "Event.h"
3. class Object
4. {
5. public:
6. Object(void);
7. virtual void Show() = 0;
8. virtual void Input() = 0;
9. virtual int Get\_Age() = 0;
10. virtual void HandleEvent(TEvent& event) = 0;
11. virtual ~Object(void);
12. };

**Файл Object.cpp:**

1. #include "Object.h"
2. Object::Object(void)
3. {
4. }
6. Object::~Object(void)
7. {
8. }

**Файл Person.h:**

1. #pragma once
2. #include "Object.h"
3. #include <iostream>
4. #include <string>
5. using namespace std;
7. class Person :
8. public Object
9. {
10. protected:
11. string name;
12. int age;
14. public:
15. Person(void);
16. public:
17. virtual ~Person(void);
18. void Show();
19. void Input();
20. Person(string, int);
21. Person(const Person&);
23. string Get\_Name() { return name; }
24. int Get\_Age() { return age; }
26. void Set\_Name(string);
27. void Set\_Age(int);
29. void HandleEvent(TEvent& event);
31. Person& operator=(const Person&);
32. };

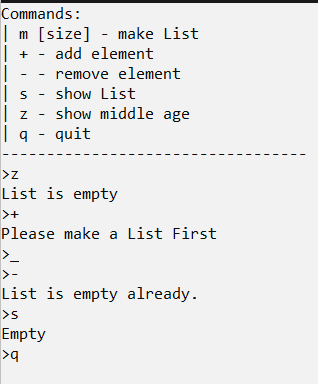
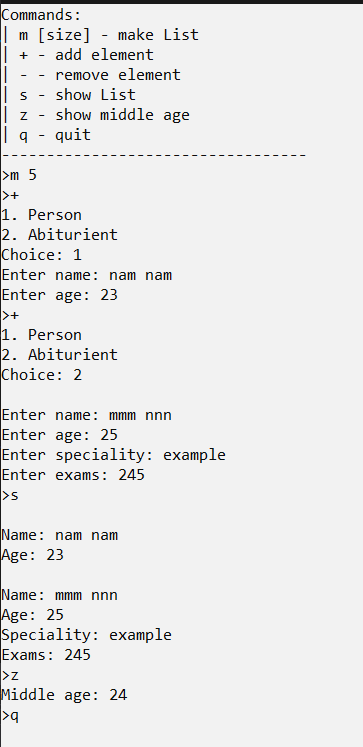
**Файл Person.cpp:**

1. #include "Person.h"
3. Person::Person(void)
4. {
5. name = "";
6. age = 0;
7. }
8. Person::Person(string n, int a)
9. {
10. name = n;
11. age = a;
12. }
13. Person::Person(const Person& a)
14. {
15. name = a.name;
16. age = a.age;
17. }
18. Person::~Person(void)
19. {}
21. void Person::Set\_Name(string n)
22. {
23. name = n;
24. }
25. void Person::Set\_Age(int a)
26. {
27. age = a;
28. }
30. Person& Person::operator=(const Person& a)
31. {
32. if (&a == this) return \*this;
33. name = a.name;
34. age = a.age;
35. return \*this;
36. }
38. void Person::Show()
39. {
40. cout << "\nName: " << name;
41. cout << "\nAge: " << age << endl;
42. }
44. void Person::Input()
45. {
46. cout << "Enter name: "; cin.ignore(); getline(cin, name);
47. cout << "Enter age: "; cin >> age;
48. cin.get();
49. }
51. void Person::HandleEvent(TEvent& event)
52. {
53. if (event.what == evMessage)
54. {
55. switch (event.command)
56. {
57. case cmName:
58. {
59. cout << "Name: " << Get\_Name() << endl;
60. break;
61. }
62. }
63. }
64. }

**Файл main.cpp:**

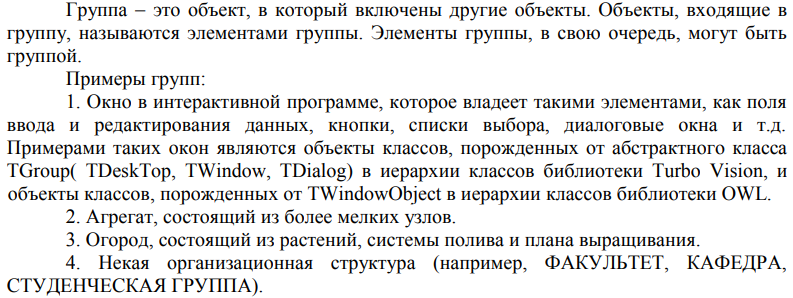
1. #include "Object.h"
2. #include "Person.h"
3. #include "Abiturient.h"
4. #include "List.h"
5. #include "Dialog.h"
6. #include <iostream>
7. using namespace std;
9. int main()
10. {
11. cout << "Commands: \n";
12. cout << "| m [size] - make List\n";
13. cout << "| + - add element\n";
14. cout << "| - - remove element\n";
15. cout << "| s - show List\n";
16. cout << "| z - show middle age\n";
17. cout << "| q - quit\n";
18. cout << "----------------------------------\n";
19. Dialog D;
20. D.Execute();
22. return 0;
23. }

**Скриншоты результатов работы программы**

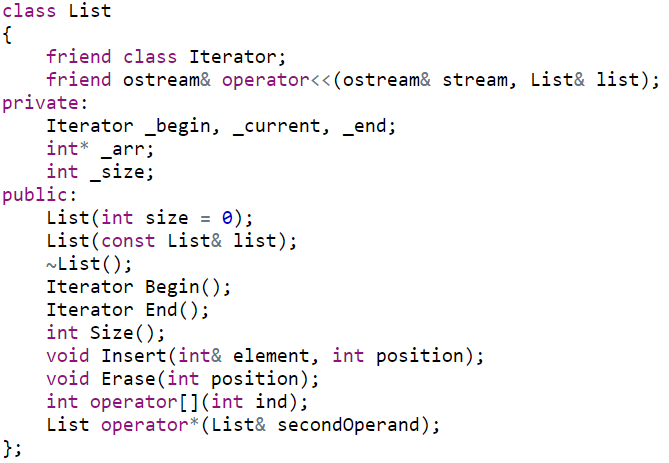
****

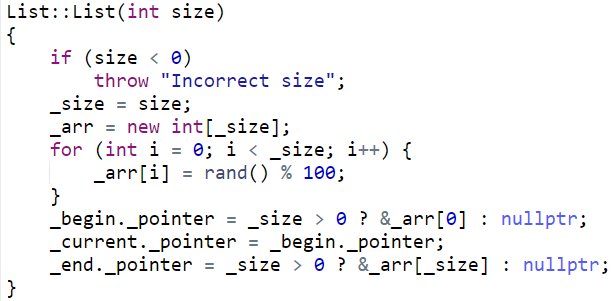
**Ответы на вопросы**

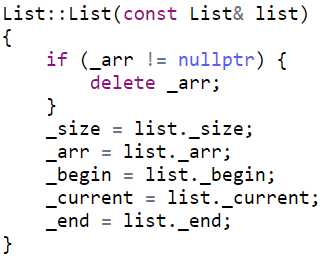


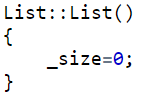
1. 



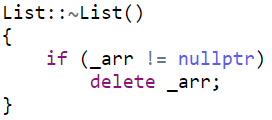
2. 

3. 

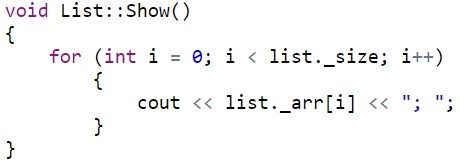




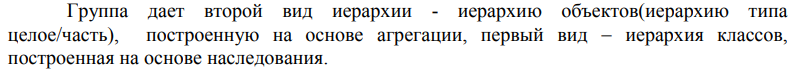


4. 



5. 



6. 

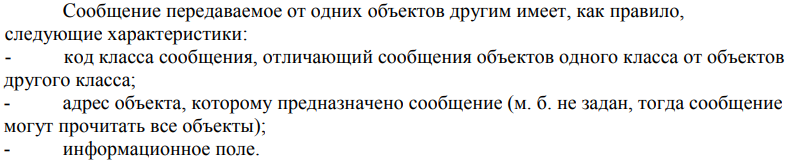


7. Для того, чтобы хранить ссылку на дочерний класс.

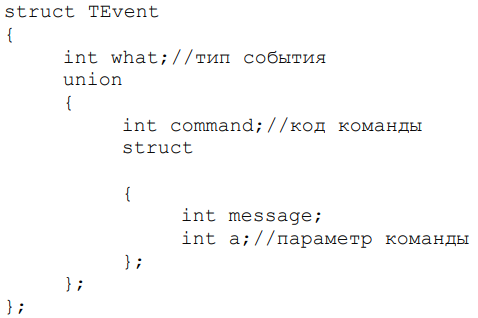


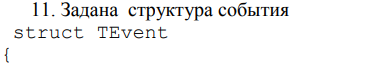
8. Событие – это пакеты информации, которыми обмениваются объекты и которые создаются объектно-ориентированной средой в ответ на те или иные действия пользователя.

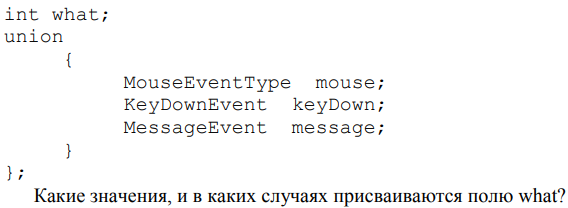


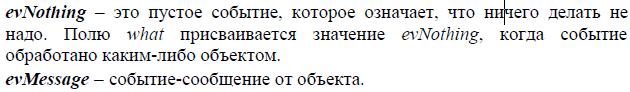
9. 

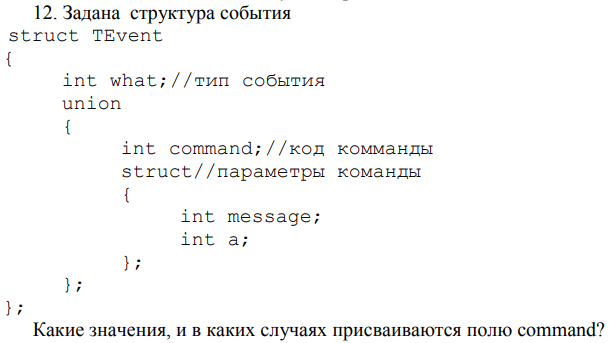


10. 

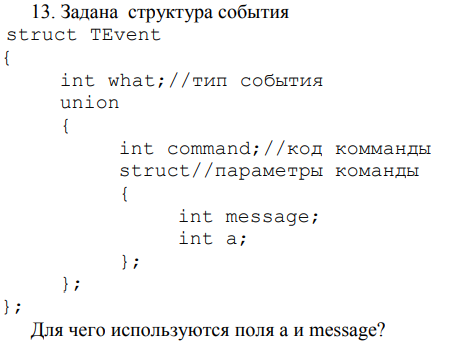




11. 

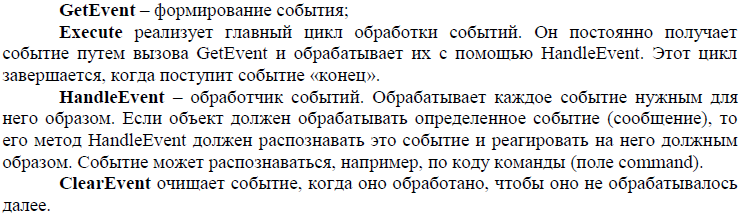


12. Код команды, которую необходимо выполнить при появлении некого события (через ветвление switch).

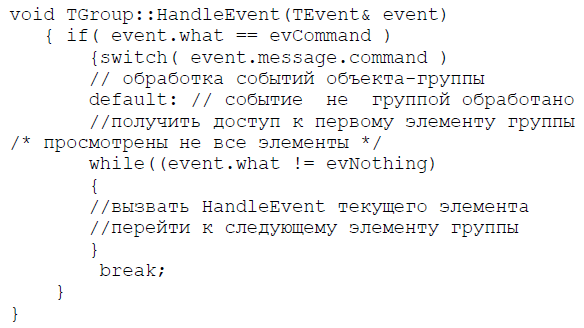


13. В поле a передаётся необходимый в дальнейшем параметр, а поле message содержит константу из переменной command.



14. 

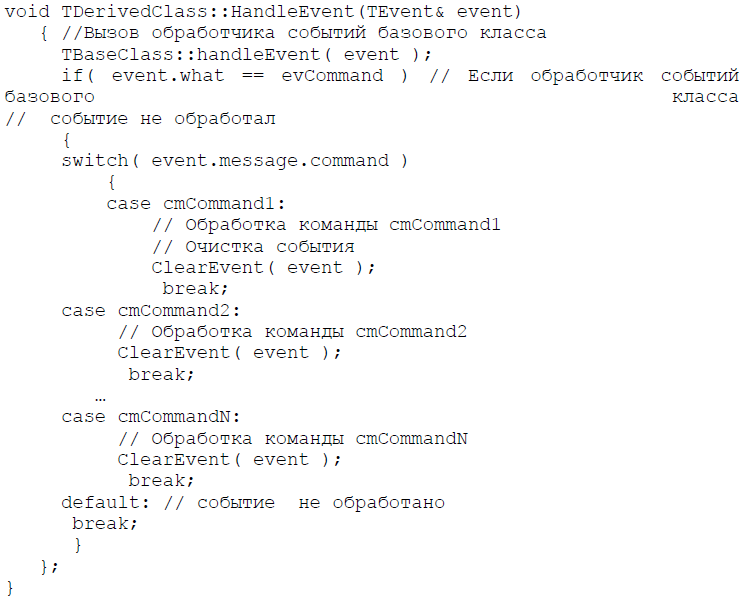


15. 



16. 



17. 



18. Формирование события.



19. Для регулирования работы программы событий: если равна 0, то программа продолжает работу, если равна 1, то завершает работу.



20. Valid отвечает за состояние параметра EndState: если параметр равен 0, цикл в методе Execute() продолжается, а если параметр равен 1, то происходит выход из цикла и остановка программы.