ТЕСТОВЫЕ ВОПРОСЫ

по курсу «Программирование на языке С++ »

тест содержит 170 вопросов

Тема 2. Основные принципы и понятия языка С++. Консольные и визуальные приложения. Основные встроенные типы данных. Переменные, константы.

1. Файлы с текстами программ на языке С++ имеют расширение
   1) **.h, *.hpp, *.c* или **.cpp*;
   2) **.txt* или **.doc*;
   3) **.obj* или **.lib*.

2. Заголовочные файлы (с расширением **.h* или **.hpp*) в языке С++ используются для
   1) объявления в них переменных программы;
   2) раздельной компиляции модулей программы;
   3) хранения массивов данных программы.

3. Заголовочные файлы (с расширением **.h* или **.hpp*) в языке С++ подключаются к компилируемому файлу
   1) с помощью директивы *#include*
   2) с помощью директивы *#inpute*
   3) с помощью директивы *#insert*

4. Точкой входа в программу на языке С++ (из перечисленных) является функция
   1) *begin( )*
   2) *start( )*
   3) *main( )*

5. Один и тот же заголовочный файл (с расширением **.h* или **.hpp*) можно подключать
   1) только к одному модулю программы;
   2) только к двум модулям программы
   3) к любому количеству модулей программы;

6. Программа на языке С++ начинает свою работу
   1) с первой строки первого модуля программы;
   2) с функции *main( ) или WinMain( )*;
   3) с произвольного места, помеченного программистом директивой *#begin*.

7. Интегрированная среда разработчика С++ Builder позволяет создавать
   1) только консольные приложения;
   2) только приложения с визуальным интерфейсом;
   3) и консольные и визуальные приложения – по выбору программиста.

8. В языке С++ символьные литеральные константы представляют собой
   1) одиночный символ, заключённый в апострофы, например *'w', 'g'* или *'7'*;
   2) последовательность символов, заключённую в двойные кавычки, например "Это строка";
   3) последовательность цифр, не начинающуюся с 0, например *23, 2003*.

9. В языке С++ строковые литеральные константы представляют собой
   1) одиночный символ, заключённый в апострофы, например *'w', 'g'* или *'7'*;

2) последовательность символов, заключённую в двойные кавычки, например *"Это строка"*;
3) последовательность цифр, не начинающуюся с 0, например *23, 2003*.

10. В языке C++ целые десятичные литеральные константы представляют собой
    1) последовательность цифр от 0 до 7, начинающаяся с 0, например *011* или *0147*;
    2) последовательность шестнадцатеричных цифр 0-9 и A-F, перед которой стоит 0X или 0x, например *0xffff*;
    3) последовательность цифр, не начинающуюся с 0, например *23, 2003*.

11. В языке C++ целые восьмеричные литеральные константы представляют собой
    1) последовательность цифр от 0 до 7, начинающаяся с 0, например *011* или *0147*;
    2) последовательность шестнадцатеричных цифр 0-9 и A-F, перед которой стоит 0X или 0x, например *0xffff*;
    3) последовательность цифр, не начинающуюся с 0, например *23, 2003*.

12. В языке C++ целые шестнадцатиричные литеральные константы представляют собой
    1) последовательность цифр от 0 до 7, начинающаяся с 0, например *011* или *0147*;
    2) последовательность шестнадцатеричных цифр 0-9 и A-F, перед которой стоит 0X или 0x, например *0xffff*;
    3) последовательность цифр, не начинающуюся с 0, например *23, 2003*.

13. В языке C++ вещественные десятичные литеральные константы представляют собой
    1) последовательность цифр, разделённых точкой, не начинающуюся с 0, например *23.0* или *3.14*;
    2) последовательность цифр, разделённых запятой, не начинающуюся с 0, например *23,0* или *3,14*;
    3) последовательность цифр, содержащих мантиссу и показатель степени числа 10, например, *3e10, 5.12E-6*.

14. В языке C++ литеральная константа *3e02* задаёт число
    1) 0.03;
    2) 30;
    3) 300;

15. В языке C++ литеральная константа *3e-02* задаёт число
    1) 0.3;
    2) 30;
    3) 0.03;

16. В языке C++ встроенный тип данных «*char*» предназначен для хранения
    1) целых чисел или символов;
    2) вещественных чисел;
    3) символов;

17. В языке C++ встроенный тип данных «*int*» предназначен для хранения
    1) символов;
    2) положительных и отрицательных целых чисел;
    3) вещественных чисел.

18. В языке C++ встроенный тип данных «*double*» предназначен для хранения
    1) символов;

2) вещественных чисел;
3) целых чисел.

19. В языке С++ идентификаторы *Index, INDEX* и *index* обозначают
1) одну и ту же переменную;
2) две различных переменных;
3) три различных переменных.

20. Именно такое объявление переменной в языке С++ является НЕ правильным
1) int x;
2) int x = 0;
3) x: int;

21. Именно такое объявление переменной в языке С++ является НЕ правильным
1) int x1;
2) int 1x;
3) int x1 = 1;

22. В языке С++ под типизированными константами понимаются
1) литеральные константы;
2) переменные, значения которых нельзя изменить;
3) параметры компилятора.

23. В языке С++ типизированные константы объявляются следующим образом:
1) const inx x = 1;
2) int x = 1;
3) int x = 1 (const);

24. В языке С++ присвоить значение типизированной константе
1) нельзя;
2) можно только при её объявлении;
3) можно в любом месте программы.

25. В языке С++ основное отличие переменных от типизированных констант состоит в том, что
1) типизированной константе присвоить значение можно только при её объявлении, переменной – в любом месте программы;
2) переменная существует в памяти машины, типизированная константа не существует;
3) типизированную константу можно инициализировать значением при создании, переменную нельзя.

Тема 3. Основные операции языка С++. Консольный ввод и вывод. Массивы.

1. В языке С++ результатом выполнения операции *4 * 5* будет число
1) 20;
2) 20.0
3) 0.

2. В языке С++ результатом выполнения операции *4.0 * 5* будет число
1) 20;

2) 20.0
3) 0.

3. В языке С++ результатом выполнения операции *5 / 2* будет число
1) 3
2) 2
3) 2.5

4. В языке С++ результатом выполнения операции *5 .0 / 2* будет число
1) 3
2) 2
3) 2.5

5. В языке С++ результатом выполнения операции *1 / 2* будет число
1) 0
2) 1
3) 0.5

6. В языке С++ результатом выполнения операции *1.0 / 2* будет число
1) 0
2) 1
3) 0.5

7. В языке С++ результатом выполнения операции *5 % 2* будет число
1) 1
2) 2
3) 3

8. В результате выполнения программы
```
int x, y;
x = 0;
y = 0;
x = y++;
```
переменная х получит значение
1) 0;
2) 1;
3) 2;

9. В результате выполнения программы
```
int x, y;
x = 0;
y = 0;
x = y++;
```
переменная у получит значение
1) 0;
2) 1;
3) 2;

10. В результате выполнения программы
```
int x, y;
x = 0;
y = 0;
x = ++y;
```

переменная y получит значение

1) 0;
2) 1;
3) 2;

11. В результате выполнения программы

```
int x, y;
x = 0;
y = 0;
x = ++y;
```

переменная y получит значение

1) 0;
2) 1;
3) 2;

12. В результате выполнения программы

```
int x, y;
x = 10;
y = 10;
x = --y;
```

переменная y получит значение

1) 0;
2) 9;
3) 10;

13. В результате выполнения программы

```
int x, y;
x = 10;
y = 10;
x = y--;
```

переменная y получит значение

1) 0;
2) 10;
3) 9;

14. Именно таким образом в языке C++ объявляется двумерный массив *xx*

1) `int xx[10, 10];`
2) `int xx[10][10];`
3) `xx : int[10][10];`

15. Массивы в языке C++ могут быть

1) только одномерными;
2) одномерными или двумерными;
3) с любым количеством измерений;

16. Индексация массивов в языке C++ начинается

1) с единицы
2) с нуля
3) с любого индекса, определяемого программистом

17. Если одномерный массив в языке C++ состоит из N элементов, то его индекс может принимать значения

1) от 0 до N-1
2) от 1 до N
3) от 0 до N

18. Для вывода информации на консоль в стандартном языке С++ используется функция
1) output( );
2) write( );
3) printf( );

19. В стандартном языке С++ для хранения текстовых строк используется
1) массив переменных типа *char*;
2) специальный строковый тип данных *string*;
3) массив переменных типа *string*;

20. В результате выполнения программы
```
int x, y;
x=10;
y=20;
printf("x = %d", x);
```
на консоль будет выведена строка:
1) x = 10
2) x = 20
3) x = 0

21. В результате выполнения программы
```
int x, y;
x=10;
y=20;
printf("y = %d", x);
```
на консоль будет выведена строка:
1) y = 10
2) x = 20
3) x = 10

22. В результате выполнения программы
```
int x, y;
x=10;
y=20;
printf("y = %d", y);
```
на консоль будет выведена строка:
1) y = 10
2) y = 20
3) x = 10

23. Количество символов в строке в языке С++ определяется
1) положением в массиве символов числа 0 (терминального нуля);
2) размером массива, заданным при его объявлении;
3) фиксированным числом – 255.

24. Максимальная длина строки в языке С++
1) не может превышать 255 символов
2) не может превышать 1024 символа

3) не ограничена

25. Признаком конца строки в языке С++ является
1) код 0 в одном из элементов массива символов (терминальный нуль);
2) код *ff* (восьмиразрядная единица);
3) специальный символ &.

Тема 4. Управляющие конструкции языка С++: условные операторы и циклы.

1. С точки зрения языка С++ выражение является истинным, если
1) это выражение равно 0
2) это выражение не равно 0
3) это выражение равно 1

2. С точки зрения языка С++ выражение является ложным, если
1) это выражение равно 0
2) это выражение не равно 0
3) это выражение равно 1

3. В результате выполнения программы

```
int x, y;
x = 0;
y = 0;
if (x)
{
y = 1;
)
```

переменная у получит значение
1) 0
2) 1
3) -1

4. В результате выполнения программы

```
int x, y;
x = 0;
y = 0;
if (!x)
{
y = 1;
)
```

переменная у получит значение
1) 0
2) 1
3) -1

5. В результате выполнения программы

```
int x, y;
x = 1;
y = 1;
if (!x)
{
```

```
y = 0;
)
```

переменная y получит значение

1) 0
2) 1
3) -1

6. В результате выполнения программы

```
int x, y;
x = 1;
y = 1;
if (x)
{
y = 0;
)
```

переменная y получит значение

1) 0
2) 1
3) -1

7. В результате выполнения программы

```
int x, y;
x = 1;
y = 1;
while (x < 1)
{
x = x + 1;
y = y + 1;
}
```

переменная y получит значение

1) 1;
2) 2;
3) 0;

8. В результате выполнения программы

```
int x, y;
x = 1;
y = 1;
while (x < 2)
{
x = x + 1;
y = y + 1;
}
```

переменная y получит значение

1) 1;
2) 2;
3) 0;

9. В результате выполнения программы

```
int x, y;
x = 1;
y = 1;
```

```
do
{
x = x + 1;
y = y + 1;
}
while (x < 1);
```

переменная y получит значение

1) 1
2) 2
3) 3

10. В результате выполнения программы

```
int x, y;
x = 1;
y = 1;
do
{
x = x + 1;
y = y + 1;
}
while (x < 2);
```

переменная y получит значение

1) 1
2) 2
3) 3

11. В результате выполнения программы

```
int x, y;
y = 1;
for(x=0; x<3; x++)
{
y=y * 2;
}
```

переменная y получит значение

1) 2
2) 4
3) 6

12. В результате выполнения программы

```
int x, y;
y = 1;
for(x=0; x<3; x++)
{
y=y * 2;
}
```

переменная x получит значение

1) 1
2) 2
3) 3

13. В языке C++ существует специальный оператор прерывания циклов *break*. Он служит для того, чтобы

1) досрочно прекратить выполнение содержащего его ближайшего цикла *while, do ... while* или *for или* условного оператора *switch*.
2) досрочно прекратить выполнение текущей итерации содержащего его ближайшего цикла *while, do ... while* или *for*.
3) досрочно завершить программу.

14. В языке C++ существует специальный оператор прерывания циклов *continue*. Он служит для того, чтобы
1) досрочно прекратить выполнение содержащего его ближайшего цикла *while, do ... while* или *for или* условного оператора *switch*.
2) досрочно прекратить выполнение текущей итерации содержащего его ближайшего цикла *while, do ... while* или *for*.
3) досрочно завершить программу.

15. Основное отличие операторов прерывания циклов *break* и *continue* состоит в том, что
1) оператор *break* прерывает выполнение содержащего его цикла, оператор *continue* только текущей итерации содержащего его цикла;
2) оператор *continue* прерывает выполнение содержащего его цикла, оператор *break* только текущей итерации содержащего его цикла;
3) оператор *break* может использоваться в циклах *for*, оператор *continue* не может;

Тема 5: Функции в языке C++. Область действия переменных и связанные с ней понятия. Модули программы.

1. Основным типом подпрограмм в языке C++ является
1) процедура
2) функция
3) оператор повторений

2. Оператор *return* в языке C++ служит для
1) возвращения функцией значения и прекращения её работы
2) прекращения функцией работы без возвращения ею значения
3) возвращения функцией значения без прекращения её работы

3. Если в функции на языке C++ отсутствует оператор *return*, то такая функция
1) не будет возвращать значения
2) будет возвращать значение 1
3) будет возвращать значение 0

4. Прототипом функции называется
1) словесное описание действий функции
2) перечень переменных, объявленных в функции
3) заголовок функции без её тела, оканчивающийся символом ‘*;*’

5. В языке C++
1) имеется понятие «вложенной» функции
2) отсутствует понятие «вложенной» функции

3) можно описывать вложенные функции при установке соответствующих директив компилятора

6. Если функции имеют одинаковое имя, но разное количество или тип параметров, то такие функции называются
1) вложенными
2) глобальными
3) перегруженными

7. Перегруженные функции применяются тогда, когда
1) необходимо смоделировать вложенность функций
2) функция должна выполнять различные действия в зависимости от типа и количества её параметров
3) нужно объявить глобальную функцию

8. Если функции отличаются типом или количеством параметров, то
1) их можно перегружать
2) их нельзя перегружать
3) возможность их перегрузки зависит от настроек директив компилятора

9. Если функции отличаются только типом возвращаемого значения, то
1) их можно перегружать
2) их нельзя перегружать
3) возможность их перегрузки зависит от настроек директив компилятора

10. В языке C++ областью действия глобальной переменной по умолчанию является
1) вся программа
2) тот модуль программы, в котором она объявлена
3) та функция, в которой она объявлена

11. В языке C++ областью действия локальной переменной по умолчанию является
1) вся программа
2) тот модуль программы, в котором она объявлена
3) та функция, в которой она объявлена

12. Для того, чтобы распространить область действия переменной на всю программу, применяется спецификатор переменных
1) *auto*
2) *volatile*
3) *extern*

13. Для того, чтобы иметь возможность вызывать функции, описанные в одном модуле программы на языке C++, из другого модуля нужно
1) создать заголовочный файл, поместить в него прототипы функций и подключить этот заголовочный файл к вызывающему модулю
2) создать заголовочный файл, поместить в него локальные переменные функций и подключить этот заголовочный файл к вызывающему модулю
3) скопировать описание функций из одного модуля в другой

14. Функции, описанные в одном модуле программы на языке C++
1) всегда могут быть вызваны из другого модуля
2) никогда не могут быть вызваны из другого модуля

3) могут быть вызваны из другого модуля при условии подключения к нему соответствующего заголовочного файла

15. Глобальная переменная, описанная в одном модуле программы на языке С++
1) всегда может быть использована в другом модуле программы;
2) никогда не может быть использована в другом модуле программы;
3) может быть использована в другом модуле программы при условии использования спецификатора *extern*

Тема 6: Указатели. Динамическая память.

1. Основное отличие динамического размещения данных от статического состоит в том, что
1) статические данные размещаются на диске, динамические – в памяти машины
2) статические данные размещаются в момент старта программы, динамические – при её выполнении
3) динамические данные размещаются в момент старта программы, статические - при её выполнении

2. Адресом в памяти машины называется
1) порядковый номер ячейки памяти
2) ссылка на данные в глобальной сети Интернет
3) имя переменной, расположенной в памяти

3. Указателем называется
1) переменная, хранящая в качестве значения какой-либо адрес в памяти машины
2) любая локальная переменная
3) любая глобальная переменная

4. В языке С++ с помощью операции взятия адреса можно получить указатель
1) на любую переменную
2) только на локальную переменную
3) только на глобальную переменную

5. Имея указатель, в языке С++
1) всегда можно получить хранящееся по этому указателю значение переменной
2) можно получить хранящееся по этому указателю значение переменной, только если она глобальная
3) можно получить хранящееся по этому указателю значение переменной, только если она локальная

6. В языке С++ имя массива без индекса эквивалентно
1) первому элементу массива;
2) указателю на первый элемент массива;
3) последнему элементу массива

7. В языке С++
1) никогда нельзя разместить в памяти одномерный динамический массив
2) всегда можно разместить в памяти одномерный динамический массив

3) можно разместить в памяти одномерный динамический массив только в случае, если его значениями являются целые числа

8. В языке C++
1) никогда нельзя разместить в памяти двумерный динамический массив
2) всегда можно разместить в памяти двумерный динамический массив
3) можно разместить в памяти двумерный динамический массив только в случае, если его значениями являются целые числа

9. Если в функцию в качестве параметра передана переменная по значению, то изменение этой переменной внутри тела функции
1) приведёт к её изменению и вне тела функции;
2) не приведёт к её изменению вне тела функции;
3) приведёт к её изменению вне тела функции, но только при соответствующих настройках директив компилятора

10. Если в функцию в качестве параметра передан указатель на переменную, то изменение этой переменной внутри тела функции
1) приведёт к её изменению и вне тела функции;
2) не приведёт к её изменению вне тела функции;
3) приведёт к её изменению вне тела функции, но только при соответствующих настройках директив компилятора

11. Если в функцию в качестве параметра передана переменная по ссылке, то изменение этой переменной внутри тела функции
1) приведёт к её изменению и вне тела функции;
2) не приведёт к её изменению вне тела функции;
3) приведёт к её изменению вне тела функции, но только при соответствующих настройках директив компилятора

12. В языке C++ в функцию в качестве параметров
1) нельзя передавать указатели на переменные
2) можно передавать указатели на переменные
3) можно передавать указатели на переменные, только если это массивы

13. В языке C++ возвращаемое значение функции
1) может быть указателем
2) не может быть указателем
3) может быть указателем, только если это указатель на массив

14. В языке C++ в памяти можно разместить многомерный динамический массив
1) только если его размерность не превышает 2
2) только если его размерность не превышает 3
3) в любом случае

15. В языке C++ размер памяти, занимаемый указателем на тип *double* (то есть переменной типа *double**)
1) больше, чем размер памяти, занимаемый указателем на тип char
2) меньше, чем размер памяти, занимаемый указателем на тип char
3) такой же, как и размер памяти, занимаемый указателем на тип char

Тема 7: Файлы двоичные и текстовые. Доступ к файлам на языке С++.

1. Процедура открытия файла заключается в
    1) создании переменной типа *FILE*
    2) блокировке операций с файлом со стороны других приложений
    3) создании переменной типа *FILE* и связывании её с конкретным файлом на диске

2. Процедура открытия файла в языке С++ выполняется функцией
    1) assign( );
    2) fopen( );
    3) openfile( );

3. С помощью функции *fopen( )* можно открыть файл
    1) только на чтение
    2) только на запись
    3) на чтение и на запись

4. С помощью функции *fopen( )* можно открыть файл
    1) только в текстовом режиме
    2) только в двоичном режиме
    3) в двоичном или текстовом режиме

5. Файловым указателем места называется
    1) переменная типа *FILE**
    2) переменная, содержащая адрес байта в файле, начиная с которого будет осуществляться операция чтения или записи
    3) переменная, которая будет записана в файл

6. Отличие файла, открытого в текстовом режиме, от файла, открытого в двоичном режиме, состоит в том, что
    1) для файла, открытого в текстовом режиме, становятся доступны специальные функции чтения и записи текста
    2) в файл, открытый в двоичном режиме, нельзя записать текст
    3) файл, открытый в двоичном режиме, может содержать только числа, файл, открытый в текстовом режиме, может содержать только текст

7. В языке С++ программист может сам задавать положение файлового указателя места. Это делается функцией
    1) *setfile( );*
    2) *fpointer( );*
    3) *fseek( );*

8. Функция *fseek( )* может установить файловый указатель места
    1) только в начало файла;
    2) только в конец файла;
    3) в любое место файла;

9. Если программисту требуется записать в файл одномерный массив из 10 целых чисел, то ему необходимо
    1) обязательно вызвать функцию *fwrite( )* 10 раз, для каждого элемента массива;
    2) вызвать функцию *fwrite( )* 1 раз, записав сразу весь массив;

3) вызвать функцию *fwrite( )* 2 раза, записав сначала элементы массива, а затем значение указателя на массив;

10. В результате выполнения программы

```
FILE *f1;
f1=fopen("data.txt", "rt");
```

1) файл *data.txt* будет открыт на чтение и запись в текстовом режиме
2) файл *data.txt* будет открыт на чтение в текстовом режиме
3) файл *data.txt* будет открыт на чтение в двоичном режиме

11. В результате выполнения программы

```
FILE *f1;
f1=fopen("data.txt", "wt");
```

1) файл *data.txt* будет открыт на чтение и запись в текстовом режиме
2) файл *data.txt* будет открыт на запись в текстовом режиме
3) файл *data.txt* будет открыт на чтение в двоичном режиме

12. В результате выполнения программы

```
FILE *f1;
f1=fopen("data.txt", "rb");
```

1) файл *data.txt* будет открыт на чтение и запись в текстовом режиме
2) файл *data.txt* будет открыт на чтение в текстовом режиме
3) файл *data.txt* будет открыт на чтение в двоичном режиме

13. В результате выполнения программы

```
FILE *f1;
f1=fopen("data.txt", "at");
```

1) файл *data.txt* будет открыт на запись в конец файла в текстовом режиме
2) файл *data.txt* будет открыт на чтение в текстовом режиме
3) файл *data.txt* будет открыт на чтение в двоичном режиме

14. В результате выполнения программы

```
FILE *f1;
f1=fopen("data.txt", "w+t");
```

1) файл *data.txt* будет открыт на чтение и запись в текстовом режиме
2) файл *data.txt* будет открыт на чтение в текстовом режиме
3) файл *data.txt* будет открыт на чтение в двоичном режиме

15. В результате выполнения программы

```
FILE *f1;
f1=fopen("data.txt", "w+b");
```

1) файл *data.txt* будет открыт на чтение и запись в двоичном режиме
2) файл *data.txt* будет открыт на чтение в текстовом режиме
3) файл *data.txt* будет открыт на чтение в двоичном режиме

Тема 8. Типы данных, определяемые пользователем. Переименование типов, перечислимые типы, структуры.

1. Перечислимые типы служат для представления переменных, которые могут принимать значения из заданного набора

1) целых именованных констант
2) вещественных именованных констант
3) символьных именованных констант

2. По умолчанию, в перечислимых типах первая из набора именованных констант представляется значением
1) 1
2) 2
3) 0

3. При объявлении перечислимых типов
1) можно указать значения именованных констант явным образом
2) нельзя указать значения именованных констант явным образом
3) можно указать значения именованных констант явным образом, только если первая будет иметь значение 0

4. При объявлении перечислимых типов различные именованные константы
1) обязательно будут иметь различные числовые значения
2) могут иметь одинаковые значения
3) могут иметь одинаковые значения, только если первая будет иметь значение 0

5. При объявлении перечислимых типов именованные константы
1) могут иметь отрицательные числовые значения
2) не могут иметь отрицательных числовых значений
3) могут иметь отрицательные числовые значения, только если первая из них имеет отрицательное числовое значение

6. Элементы структур располагаются в памяти компьютера
1) последовательно, один за другим
2) начинаясь с одного адреса памяти, перекрывая друг друга
3) только на жёстком диске

7. Элементы объединений располагаются в памяти компьютера
1) последовательно, один за другим
2) начинаясь с одного адреса памяти, перекрывая друг друга
3) только на жёстком диске

8. Для доступа к элементам структур в языке C++ используется операция «.» (точка). Она применяется к
1) переменным, имеющим тип «структура»
2) переменным, имеющим тип «указатель на структуру»
3) переменным, имеющим тип «перечислимый тип»

9. Для доступа к элементам структур в языке C++ используется операция «->» (стрелка). Она применяется к
1) переменным, имеющим тип «структура»
2) переменным, имеющим тип «указатель на структуру»
3) переменным, имеющим тип «перечислимый тип»

10. Массивы символов (строки)
1) могут быть элементами структур
2) не могут быть элементами структур
3) не могут быть элементами объединений

11. Целые числа
1) могут быть элементами структур
2) не могут быть элементами структур
3) не могут быть элементами объединений

12. Вещественные числа
1) могут быть элементами структур
2) не могут быть элементами структур
3) не могут быть элементами объединений

13. К элементам структур можно обратиться
1) только из функций, описанных в том же модуле, что и структура
2) только из функции *main( )*
3) из любой функции программы

14. Элементы структур доступны
1) только для чтения
2) только для записи
3) для чтения и для записи

15. В языке С++ элементы структур
1) могут, в свою очередь, также быть структурами
2) не могут быть структурами
3) не могут быть объединениями

Тема 9. Визуальные приложения в Windows. Создание визуальных приложений в среде C++ Builder

1. В операционной системе *Windows* реализуется многозадачность
1) основанная на потоках
2) основанная на процессах
3) основанная на потоках и процессах

2. В операционной системе *Windows* реализуется многозадачность, основанная на потоках и процессах. При этом под процессом понимается
1) отдельная выполняемая программа
2) отдельная часть исполняемого кода программы
3) сеанс обращения к жёсткому диску компьютера

3. В операционной системе *Windows* реализуется многозадачность, основанная на потоках и процессах. При этом под потоком понимается
1) отдельная выполняемая программа
2) отдельная часть исполняемого кода программы
3) сеанс обращения к жёсткому диску компьютера

4. При многозадачности, основанной на потоках,
1) две или более программы могут выполняться параллельно
2) отдельные потоки внутри процесса могут выполняться параллельно
3) компьютер не может иметь более двух процессоров

5. При многозадачности, основанной на процессах,
1) две или более программы могут выполняться параллельно
2) отдельные потоки внутри процесса могут выполняться параллельно
3) компьютер не может иметь более двух процессоров

6. Синхронными называются функции, исполняемые
1) в одном потоке команд
2) в разных потоках команд
3) в разных модулях программы

7. Асинхронными называются функции, исполняемые
1) в одном потоке команд
2) в разных потоках команд
3) в разных модулях программы

8. Если в программе имеется три окна, то она может содержать
1) один поток команд
2) два потока команд
3) пять потоков команд

9. Если в программе не имеется ни одного окна, то она
1) не может содержать более одного потока команд
2) не может содержать более двух потоков команд
3) может содержать любое количество потоков команд

10. Свойство *Height* визуальных компонентов задаёт
1) Ширину компонента
2) Высоту компонента
3) Цвет компонента

11. Свойство *Width* визуальных компонентов задаёт
1) Ширину компонента
2) Высоту компонента
3) Цвет компонента

12. Обработчик события *OnClick* задаёт
1) функцию, вызываемую при одном щелчке мышью на компоненте
2) функцию, вызываемую при двойном щелчке мышью на компоненте
3) функцию, вызываемую при одном нажатии клавиши на клавиатуре

13. Обработчик события *OnDblClick* задаёт
1) функцию, вызываемую при одном щелчке мышью на компоненте
2) функцию, вызываемую при двойном щелчке мышью на компоненте
3) функцию, вызываемую при одном нажатии клавиши на клавиатуре

14. Функция, вызываемая при нажатии клавиши на клавиатуре, задаётся в обработчике события
1) *OnKeyDown*
2) *OnKeyUp*
3) *OnClick*

15. Функция, вызываемая при отпускании клавиши на клавиатуре, задаётся в обработчике события

1) *OnKeyDown*
2) *OnKeyUp*
3) *OnClick*

Тема 10. Реакция на события от мыши и клавиатуры

1. Функция, вызываемая при движении указателя мыши, задаётся в обработчике события

1) *OnMouseDown*
2) *OnMouseUp*
3) *OnMouseMove*

2. Функция, вызываемая при нажатии кнопки мыши, задаётся в обработчике события

1) *OnMouseDown*
2) *OnMouseUp*
3) *OnMouseMove*

3. Функция, вызываемая при отпускании кнопки мыши, задаётся в обработчике события

1) *OnMouseDown*
2) *OnMouseUp*
3) *OnMouseMove*

4. Параметр *Shift* в функциях-обработчиках событий от мыши и клавиатуры

1) Равен значению «Истина», если при наступлении события была нажата клавиша *Shift*
2) Содержит признаки, уточняющие обстоятельства возникновения события
3) Содержит ссылку на компонент, вызвавший событие

6. Параметры *X* и *Y* в функциях-обработчиках событий от мыши

1) содержат координаты указателя мыши в системе координат монитора
2) содержат координаты указателя мыши в системе координат компонента, вызвавшего событие
3) содержат координаты указателя мыши в системе координат главной формы программы

7. Функция-обработчик события *OnKeyPress* вызывается

1) при нажатии любой клавиши клавиатуры
2) при нажатии на клавиатуре функциональных клавиш *F1 … F12*
3) при нажатии на клавиатуре любой алфавитно-цифровой клавиши

8. Функция-обработчик события *OnKeyDown* вызывается

1) при нажатии любой клавиши клавиатуры
2) при нажатии на клавиатуре функциональных клавиш *F1 … F12*
3) при нажатии на клавиатуре любой алфавитно-цифровой клавиши

9. В функции-обработчике события *OnKeyPress* параметр Key

1) не учитывает регистр и выбранный язык
2) учитывает регистр и выбранный язык
3) содержит номер клавиши на клавиатуре

10. В функции-обработчике события *OnKeyDown* параметр Key
1) не учитывает регистр и выбранный язык
2) учитывает регистр и выбранный язык
3) содержит номер клавиши на клавиатуре

Тема 11. Разработка приложений со стандартными элементами оконного интерфейса – меню, текстовым редактором и диалоговыми окнами.

1. Компонент Edit представляет собой
1) однострочный текстовый редактор
2) многострочный текстовый редактор
3) компонент выбора файла для редактирования

2. Компонент Memo представляет собой
1) однострочный текстовый редактор
2) многострочный текстовый редактор
3) компонент выбора файла для редактирования

3. Свойство *Color* компонента *Memo* задаёт
1) цвет текста в компоненте
2) цвет фона компонента
3) цвет формы, на которой расположен компонент

4. Свойство Font компонента *Memo* задаёт
1) свойства шрифта всего текста компонента
2) свойства шрифта отдельной строки текста в компоненте
3) свойства шрифта заголовка формы, на которой расположен компонент

5. В многострочном текстовом редакторе *Memo* можно задать
1) цвет текста каждой строки
2) размер шрифта каждой строки
3) цвет всех строк в компоненте

6. В многострочном текстовом редакторе *Memo* можно задать
1) цвет текста каждой строки
2) размер шрифта каждой строки
3) размер шрифта всех строк в компоненте

7. Компоненты – диалоговые окна
1) являются невизуальными
2) видны на форме в том месте, где они были помещены при конструировании формы
3) показываются на экране только при вызове специального метода *Execute( )*

8. Функция *Execute( )* в компонентах – диалоговых окнах возвращает значение
1) 0, если диалоговое окно было закрыто в режиме «отмена»
2) 0, если диалоговое окно было закрыто в режиме «ОК»
3) всегда равное 1

9. Функция *Execute( )* в компонентах – диалоговых окнах

1) всегда вызывается асинхронно
2) всегда вызывается синхронно
3) не может быть вызвана программистом

10. Выбранное имя файла в компоненте *OpenDialog* хранится в свойстве компонента
1) *FileName*
2) *NameOfFile*
3) *FileID*

Тема 12. Графический инструментарий среды C++ Builder

1. Цвет, задаваемый функцией *RGB( )*, определяется интенсивностью
1) красного, зелёного и синего цветов
2) пурпурного, жёлтого и оливкового цветов
3) чёрного и белого цветов

2. Свойство объектов *Canvas* представляет собой
1) поле, хранящее размеры компонента
2) поле, хранящее ссылку на изображение объекта
3) поле, хранящее цвет объекта

3. Функция RGB( ) позволяет выбрать цвет из доступных ей
1) 2^8 цветов
2) 2^{16} цветов
3) 2^{24} цветов

4. Набор способов заливки области графического объекта (выполняемой объектом-кистью Brush)
1) жёстко задан средой и не может быть расширен программистом
2) жёстко задан операционной системой и не может быть расширен программистом
3) может быть расширен программистом

5. Функция канвы *LineTo(x, y)*
1) чертит линию от текущего положения пера до точки x, y
2) перемещает перо в положение x, y без вычерчивания линии
3) закрашивает пиксель *(x, y)* текущим цветом

6. Функция канвы *MoveTo(x, y)*
1) чертит линию от текущего положения пера до точки x, y
2) перемещает перо в положение x, y без вычерчивания линии
3) закрашивает пиксель *(x, y)* текущим цветом

7. Параметры функции *Rectangle(x1, y1, x2, y2)* задают
1) координаты левого верхнего и правого нижнего углов прямоугольника
2) длины четырёх сторон четырёхугольника
3) длины двух сторон и двух диагоналей четырёхугольника

8. Параметры функции *Ellipse(x1, y1, x2, y2)* задают
1) координаты левого верхнего и правого нижнего углов прямоугольника, описывающего эллипс

2) координаты левого верхнего и правого нижнего углов прямоугольника, вписанного в эллипс
3) координаты центра эллипса и длины его полуосей

9. Функция *Ellipse( )*
1) рисует эллипс и не заливает его текущей кистью
2) рисует эллипс и заливает его текущей кистью
3) всегда рисует окружность, радиус которой равен меньшей полуоси соответствующего эллипса

10. Переменные класса *TBitmap* применяются для того, чтобы
1) создавать графические объекты в памяти машины
2) загружать изображение из файла
3) сохранять изображение в файл