ЭКЗАМЕНАЦИОННЫЕ ВОПРОСЫ

ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ

КАФЕДРА ВТ / ИНСТИТУТ ИТ

Для всех групп 1-го курса института ИТ

1. Аргументы, передаваемые функции по умолчанию.
2. Архитектура системы. Иерархия объектов.
3. Ввод-вывод в C++. Потоки.
4. В чем различие между ссылкой и указателем?
5. Виртуальные базовые классы.
6. Взаимодействие объектов. Три примера взаимодействия объектов.
7. Виртуальные методы. Наследование виртуальных методов.
8. В чем отличие между классом и структурой?
9. Встраиваемая функция.
10. Для чего используется ключевое слово protected?
11. Дружественная функция.
12. Дружественный класс.
13. Защищенные члены класса.
14. Жизненный цикл объекта.
15. Жизненный цикл виртуального объекта и его реализация на языке C++.
16. Инкапсуляция.
17. Имеются два способа сделать функцию встраиваемой. Что это за способы?
18. Исключительные ситуации.
19. Класс. Назначение и синтаксис описания.
20. Класс vector.
21. Класс string.
22. Какая инструкция catch перехватывает все типы исключительных ситуаций?

23. Какова основная форма конструктора копий?

24. Конструктор копии.

25. Контейнеры и итераторы.

26. Контейнер – динамический массив.

27. Какое условие является обязательным для присвоения одного объекта другому?

28. Контейнер – ассоциативный список.

29. Какой тип операций ведет к вызову конструктора копий?

30. Конструктор и деструктор объекта.

31. Класс map и multimap.

32. Можно ли адрес объекта передать функции в качестве аргумента?

33. Множественное наследование.

34. Можно ли использовать инструкцию throw, если ход выполнения программы не затрагивает инструкции, расположенные в блоке try?

35. Может ли быть инициализирован массив, память для которого выделяется динамически?

36. Наследование. Реализация наследования на языке C++.

37. Объявление элементов класса спецификацией static.

38. Объявление элементов класса спецификацией const.

39. Объявление объекта и доступ к его элементам.

40. Объединение. Назначение и синтаксис описания.

41. Объекты в качестве возвращаемого значения функции.

42. Определение адреса перегруженной функции.

43. Определение системы и три примера систем.

44. Перегрузка бинарных операторов.

45. При наследовании одного класса другим, когда вызываются конструкторы классов? Когда вызываются их деструкторы?

46. Полиморфизм.

47. Параметризированные конструкторы.

48. Программа – система.

50. Перегрузка оператора индексации массивов [ ]

51. Присвоение объектов.

52. Приведение типов.

53. Перегрузка функций.

54. Перегрузка унарных операторов.

55. Структура. Назначение и синтаксис описания.

56. Операторы new и delete.

57. Сигналы и обработчики.

58. Форматированный ввод-вывод данных.

59. Что такое родовой класс и какова его основная форма?

60. Что происходит, когда открытые члены базового класса наследуются как открытые? Что происходит, когда они наследуются как закрытые?

61. Чисто виртуальные функции и абстрактные классы.

62. Что такое родовая функция и какова ее основная форма?

63. Что такое встраиваемая функция? В чем ее преимущества и недостатки?

64. Чем действие дружественной оператор-функции отличается от действия оператор-функции — члена класса?

65. Что такое объект?

66. Что происходит с защищенным членом класса, когда класс наследуется как открытый? Что происходит, когда он наследуется как закрытый?

67. Управление доступом к элементам класса.

68. Указатели и ссылки на объект.

69. Указатель на объект производного класса

70. Управление доступом при наследовании.

71. Указатель this.

72. Шаблон класса.

73. Шаблон функции.

74. Почему следующие две перегруженные функции внутренне неоднозначны?

```
int f ( int   a );
```

```
int f ( int & a );
```

75. Почему следующая функция может не компилироваться как встраиваемая?

```
void fl ( ) {
int  i;
for( i = 0; i < 10; i++ ) cout << i;
}
```

76. Что неправильно в данном фрагменте?

```
int main ( ) {
. . . .
throw 12.23;
```

77. Что неправильно в следующем прототипе функции?

```
char * f ( char * p, int x = 0, char * q );
```

78. Что неправильно в конструкторе, показанном в следующем фрагменте?

```
class sample {
    double a, b, c;
public:
    double sample ( );
}
```

79. Правилен ли следующий фрагмент?

```
union {
    float f;
    unsigned int bits;
}
```

80. Какой будет результат после отработки данной программы?

```
#include <iostream>
#include <list>
using namespace std;
```

```
int  main ( ) {
    list < char >             lst;
    list < char > :: iterator it_lst;
    int    i;

    for ( i = 0; i < 10; i ++ ) lst.push_back ( 'A' + i );
    cout << "Size = " << lst. size ( ) << endl;

    cout << "lst: ";

    while ( ! lst.empty ( ) ) {
        it_lst = lst.end ( );
        it_lst --;
        cout << * it_lst;
        lst.pop_back ( );
    }

    return 0;
}
```

81. Какой будет результат после отработки данной программы?

```
#include <iostream>
using namespace std;

class static_func_demo {
    static int i;
public:
    static void init ( int x ) { i = x;     }
           void show ( )        { cout << i; }
};
```

```
int static_func_demo :: i;

int main ( )
{
    static_func_demo :: init ( 100 );
    static_func_demo x;
    x.show ( );

    return 0;
}
```

82. Какой будет результат после отработки данной программы?

```
#include <iostream>
using namespace std;

class samp {
    int i;
public:
         samp  ( int n ) { i = n; }
    void set_i ( int n ) { i = n; }
    int  get_i ( ) { return i; }
};

void sqr_it ( samp ob ) {
    ob.set_i ( ob.get_i ( ) * ob.get_i ( ) ) ;
    cout << ob.get_i() << "\n";
}

int main ( ) {
    samp a ( 10 );
    sqr_it (  a );

    cout <<  a.get_i ( );
```

```
    return 0;
}
```

83. Исследуйте следующую конструкцию:

```
#include <iostream>
using namespace std;

class cl_base {
    int a, b;
public:
    int  c;
    void setab ( int   i, int   j ) { a = i; b = j; }
    void getab ( int & i, int & j ) { i = a; j = b; }
};
class derived_1 : pablic  cl_base { };
class derived_2 : private cl_base { };

int main ( ) {
    derived_1  ob_1;
    derived_2  ob_2;
    int        i, j;

    // . . . . .
    return 0;
}
```

Какая из следующих инструкций правильна внутри функции main ( )?

```
A. ob_1.getab ( i, j );
B. ob_2.getab ( i, j );
C. ob_1.c = 10;
D. ob_2.c = 10;
```

84. Объясните, что в следующей программе неправильно, и исправьте ее.

```
#include <iostream>
using namespace std;

class cl_1 {
    int * p;
public:
    cl_1  ( int i );
   ~cl_1  ( ) { delete p; }
    friend int getval ( cl_1 ob );
};

cl_1 :: cl_1 ( int i ) {
    p = new int;
    if ( ! p ) {
        cout << "Error 1\п";
        exit ( 1 );
    }
    * p = i;
}

int getval ( cl_1 ob ) { return * ob.p; }

int main ( ) {
    cl_1 a ( 1 ), b ( 2 );

    cout << getval ( a ) << " " << getval ( b );
    cout << "\n";
    cout << getval ( a ) << " " << getval ( b ) ;

    return 0 ;
}
```

85. Какой будет результат после отработки данной программы?

```
#include <iostream>
using namespace std;

class A {
public: A ( ) { cout << "Constructor A\n"; }
       ~A ( ) { cout << "Destructor  A\n"; }
};
class B {
public:
    B ( ) { cout << "Constructor B\n"; }
   ~B ( ) { cout << "Destructor  B\n"; }
};
class C : public A, public B {
public:
    C ( ) { cout << "Constructor C\n"; }
   ~C ( ) { cout << "Destructor  C\n"; }
};

int main ( ) {
    C  ob;

    return 0;
}
```

86. В следующей программе имеется ошибка. Исправьте ее с помощью оператора const_cast.

```
#include <iostream>
using namespace std;

void f ( const double & i )
{
    i = 100;
```

```
}

int main ( )
{
    double x = 98.6;

    cout << x << endl;
    f ( x );
    cout << x << endl;

    return 0;
}
```

87. Ниже приведены две перегруженные функции. Покажите, как получить адрес каждой из них.

```
#include <iostream>
using namespace std;

int   dif ( int   a, int   b ) { return a – b; }
float dif ( float a, float b ) { return a – b; }

int main ( ) {

    // ??????


    return 0;
}
```

88. Какой будет результат после отработки данной программы?

```
#include <iostream>
using namespace std;

union bits {
```

```
    bits ( short n );
    void show_bits ( ) ;

    short  d;
    unsigned char c [ sizeof ( short ) ];
};

bits :: bits ( short n ) { d = n; }

void bits :: show_bits ( ) {
    int  i, j;

    for ( j = sizeof ( short ) - 1; j >= 0; j -- ) {
        cout << "Byte " << j << ":";
        for( i = 128; i; i >>= 1 )
            if ( i & c [ j ] ) cout << "1";
            else               cout << "0";
        cout << "\n";
    }
}

int main ( ) {
    bits ob ( 5 );

    ob.show_bits ( );
    return 0;
}
```

89. Какой будет результат после отработки данной программы?

```
#include <iostream>
using namespace std;

template < class T >
```

```
T & inc_value ( T & val ) {

    ++val;

    return val;
}

int main ( )
{
    int  x = 64;
    char c = 64;

    x = ( int )  inc_value < int  > ( x );
    cout << x << endl;

    c = ( char ) inc_value < char > ( c );
    cout << c << endl;

    printf ( "%02X", c );

    return 0;
}
```

90. Какой будет результат после отработки данной программы?

```
#include <iostream>
using namespace std;

int main ( ) {
    union {
        unsigned char bytes [ 4 ];
        int           value;
    };
```

```
    int i;
    value = 128;

    for ( i = 3; i >= 0; i -- )
        cout << ( int ) bytes [ i ] << " ";

    return 0;
}
```

91. Какой будет результат после отработки данной программы?

```
#include <iostream>
using namespace std;

class samp {
    int i;
public:
         samp   ( int n ) { i = n; }
    void set_i  ( int n ) { i = n; }
    int  get_i  ( )       { return i; }
};

void sqr_it ( samp * ob ) {

    ob -> set_i ( ob -> get_i ( ) * ob -> get_i ( ) );

    cout << ob -> get_i ( ) << "\n";
}

int main ( ) {
    samp a ( 10 );

    sqr_it ( & a );
    cout << a.get_i ( );
```

```
    return 0;
}
```

92. Какой будет результат после отработки данной программы?

```
#include <iostream>
using namespace std;

template < class T1 >
void PrintArray ( const T1 * array, const int count )
{
    for ( int i = 0; i < count; i++ )
        cout << array [ i ] << " ";
    cout << endl;
}

int main ( )
{
    const int aCount = 5;
    const int bCount = 7;
    const int cCount = 6;
    int    a [ aCount ] = { 1,2,3,4,5 };
    double b [ bCount ] = { 1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7 };
    char   c [ cCount ] = "HELLO";

    cout << "Array a:" << endl;
    PrintArray ( a, aCount );

    cout << "Array b:" << endl;
    PrintArray ( b, bCount );

    cout << "Array c:" << endl;
```

```
    PrintArray ( c, cCount );

    return 0;
}
```

93. Дана следующая программа, переделайте все соответствующие обращения к членам класса так, чтобы в них явно присутствовал указатель this.

```
#include <iostream>
using namespace std;

class cl_1 {
    int a, b;
public:
    cl_1      ( int n, int m ) { a = n; b = m; }
    int  add  ( )              { return a + b; }
    void show ( );
};

void cl_1 :: show ( ) {
    int t;
    t = add ( );
    cout << t << "\n";
}

int main() {
    cl_1  ob ( 10, 14 );
    ob.show  ( ) ;

    return 0;
}
```

94.Какой будет результат после отработки данной программы?

```
#include <iostream>
using namespace std;

class base {
public:
    base ( ) { cout << "Constructor base\n"; }
   ~base ( ) { cout << "Destructor  base\n"; }
};

class derived : public base {
public:
    derived ( ) { cout << "Constructor derived\n"; }
   ~derived ( ) { cout << "Destructor  derived\n"; }
};

int main ( ) {

    derived ob;

    return 0;
}
```

95. Что неправильно в следующей программе?

```
#include <iostream>
using namespace std;

void triple ( double & num );

int main ( ) {
    double  d = 7.0;

    triple ( & d );
    cout << d;
```

```
    return 0;
}
```

96. Какой будет результат после отработки данной программы?

```
#include <iostream>
using namespace std;

union swapbytes {
      unsigned char  c [ 2 ];
      unsigned short i;
public:
    swapbytes ( unsigned short x );
    void  swp ( );
};
     swapbytes :: swapbytes ( unsigned short x ) { i = x; }
void swapbytes :: swp ( ) {
    unsigned char temp;
    temp    = c [ 0 ];
    c [ 0 ] = c [ 1 ];
    c [ 1 ] = temp;
}

int main ( ) {
    swapbytes ob ( 1 );
    ob.swp ( );
    cout << ob.i;
    return 0;
}
```

97. Какой будет результат после отработки данной программы?

```
#include <iostream>
using namespace std;
int rotate ( int i ) {
    int x;
    if ( i & 0x80000000 ) x = 1;
    else                  x = 0;
    i  = i << 1;
    i += x;
    return i;
}
 int main ( ) {
    int a;
    a = 0x80000001;
    cout << rotate ( a );
    return 0;
}
```

98. Какой будет результат после отработки данной программы?

```
#include <iostream>
using namespace std;

class cl_1 {
public:
    static int  i;

    void   seti ( int n ) { i = n;    }
    int    geti ( )       { return i; }
};

int cl_1 :: i;

int main ( ) {
    cl_1    ol, o2;
    cl_1 :: i = 100;
```

```
    cout << "o1.i: " << o1.geti ( ) << '\n';
    cout << "o2.i: " << o2.geti ( ) << '\n';
    return 0;
}
```

99. Какой будет результат после отработки данной программы?

```
#include <iostream>
using namespace std;

int rotate ( int i ) {
    int x;
    if ( i & 0x80000000 ) x = 1;
    else                  x = 0;
    i  = i << 1;
    i += x;
    return i;
}

 int main ( ) {
    int a;
    a = 0x80000000;
    cout << rotate ( a );
    return 0;
}
```

100. Какой будет результат после отработки данной программы?

```
#include <iostream>
using namespace std;

union swapbytes {
      unsigned char  c [ 2 ];
      unsigned short i;
```

```
public:
    swapbytes ( unsigned short x );
    void  swp ( );
};
     swapbytes :: swapbytes ( unsigned short x ) { i = x; }
void swapbytes :: swp ( ) {
    unsigned char temp;
    temp    = c [ 0 ];
    c [ 0 ] = c [ 1 ];
    c [ 1 ] = temp;
}

int main ( ) {
    swapbytes ob ( 256 );
    ob.swp ( );
    cout << ob.i;
    return 0;
}
```

101. Какой будет результат после отработки данной программы?

```
#include <iostream>
using namespace std;

int main()
{
    unsigned char c = 0x1A;

    c <<= 4;    // умножить  на 16
    c >>= 4;    // разделить на 16

    printf ( "c = %02X",  c );

    return 0;
}
```

102. Какой будет результат после отработки данной программы?

```
#include <iostream>
#include <typeinfo>
using namespace std;

class BaseClass {
    virtual void f ( ) { }
};
class Derivedl : public BaseClass { };
class Derived2 : public BaseClass { };

int main ( ) {
    int i ;
    BaseClass * p,   baseob;
    Derivedl    ob1;
    Derived2    ob2;

    cout << "Type - " << typeid (   i ).name ( ) << endl;
    p = & baseob;
    cout << "Type - " << typeid ( * p ).name ( ) << endl;
    p = & ob1;
    cout << "Type - " << typeid ( * p ).name ( ) << endl;
    p = & ob2;
    cout << "Type - " << typeid ( * p ).name ( ) << endl;

    return 0;
}
```

103. Какой будет результат после отработки данной программы?

```
#include <iostream>
using namespace std;
```

```
class Base {
public:
    virtual void f ( ) { }
};
class Derived: public Base {
public:
    void derived_only ( ) {
         cout << "It's object of class Derived\n";
    }
};

int main ( ) {
    Base    * bp, b_ob;
    Derived * dp, d_ob;

    bp = & b_ob;
    dp = dynamic_cast < Derived * > ( bp );
    if   ( dp ) dp -> derived_only ( );
    else        cout << "Error 1\n";

    bp = & d_ob;
    dp = dynamic_cast < Derived * > ( bp );
    if   ( dp ) dp -> derived_only ( );
    else        cout << "Error 2\n";

    return 0;
}
```

104. Что неправильно в следующем фрагменте?

```
#include <iostream>
using namespace std;
class cl1 {
    int i , j;
```

```
public:
    cl1 ( int a, int b ) { i = a; j = b; }
};
class cl2 {
    int i, j;
public:
    cl2 ( int a, int b ) { i = a; j = b; }
};

int main ( ) {
    cll x ( 10, 20 );
    cl2 y (  0,  0 );
    x = y;
    . . . . .
}
```