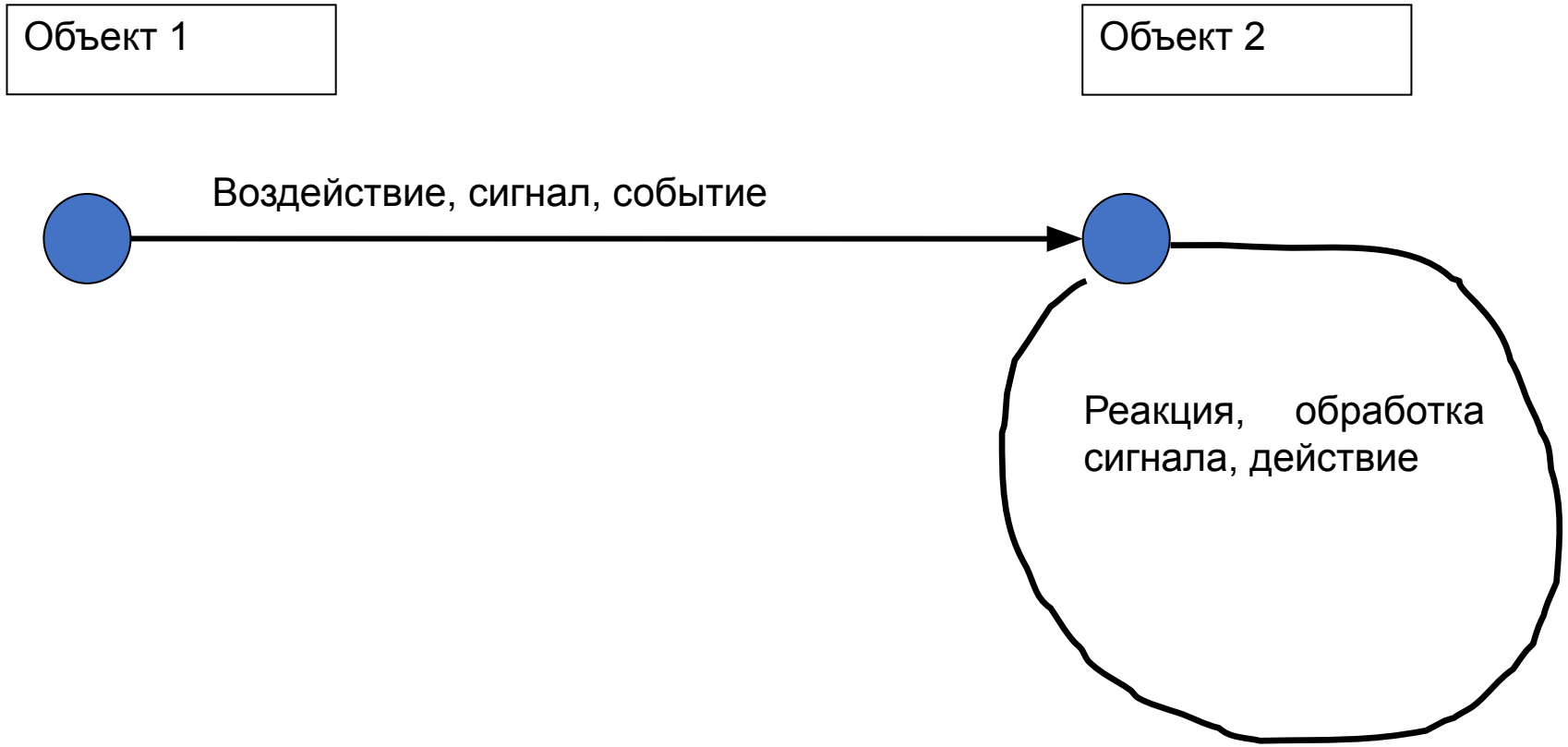
Объектно-ориентированное программирование на алгоритмическом языке C++

МИРЭА, Институт Информационных технологий, кафедра Вычислительной техники

Схема взаимодействия объектов



Пример

```
#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;
}
```

Шаблонные функции

Описание шаблона функции

template <формальные параметры > «тип» «имя функции» (список параметров)

{

// тело функции

}

формальный параметр ::= class «имя параметра» |

typename «имя параметра»

Пример шаблонной функции

```
template < class T >  T & inc_value ( T & val )   {
   ++val;   return val;
}

int main ( )
{
   int x = 0;
   x = ( int ) inc_value < int > ( x );
   cout << x << endl;
   char c = 0;
   c = ( char ) inc_value < char > ( c );
   cout << c << endl;
   return 0;
}
```

Пример шаблонной функции

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

Пример шаблонной функции

```
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"; //6-я позиция для null
    cout << "Array a:" << endl;
    PrintArray ( a, aCount ); // шаблон для integer
    cout << "Array b:" << endl;
    PrintArray ( b, bCount );  // шаблон для double
    cout << "Array c:" << endl;
    PrintArray ( c, cCount );  // шаблон для character
    return 0;
}
```

Ответ примера

```
Array a:
1 2 3 4 5
Array b:
1.1 2.2 3.3 4.4 5.5 6.6 7.7
Array c:
H E L L O
```

```
void PrintArray ( const int*,    const int );
void PrintArray ( const double*, const int );
void PrintArray ( const char*,   const int );
```

Перегрузка шаблонных функций

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

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

Перегрузка шаблонных функций

```
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"; //6-я позиция для null
    …
    cout << "Array a from 1 to 3:" << endl;
    PrintArray ( a, 2 );          // шаблон для integer
    cout << "Array b from 4 to 7:" << endl;
    PrintArray ( b, 3, 6 );       // шаблон для double
    cout << "Array c from 3 to 5:" << endl;
    PrintArray ( c, 2, 4 );       // шаблон для character
    return 0;
}
```

Перегрузка шаблонных функций

```
template < class T1 >
void PrintArray ( const T1 * array, const int count )
{…}

template < class T1 >
void PrintArray ( const T1 * array,
                  const int lowSubscript,
                  const int highSubscript )
{…}

void PrintArray ( char * array, const int count )
{
    for ( int i = 0; i < count; i++ )
        cout << array [ i ] << endl;
}
```

Шаблоны классов

```
template < class Ttype > class «имя класса»
{
   // описание класса
};
```

где - Ttype представляет имя типа, который будет задан при создании объекта класса, используя следующий формат:

«имя класса» < тип > «имя объекта»;

Здесь «тип» определяет имя типа данных, который будет обрабатываться объектом класса. Функции члены класса, автоматически являются обобщенными.

Пример шаблона класса

```
// Демонстрация класса очереди queue.
#include <iostream>
using namespace std;

const int SIZE = 100;

// Описание класса queue.
template < class Qtype > class queue {
  QType q [ SIZE ];
  int   sloc, rloc;
public:
  queue ( )  { sloc = rloc = 0; }
  void  qput ( QType i );
  QType qget ( );
};
```

Пример шаблона класса

```
template < class QType > void queue < QType > :: qput ( QType i ) {
    if ( sloc == SIZE ) {
        cout << "Очередь заполнен.\n";
        return;
    }
    Sloc ++;
    q [ sloc ] = i;
}

template < class QType > QType queue < QType > :: qget ( ) {
    if ( rloc == sloc ) {
        cout << "Очередь пуст.\n";
        return 0;
    }
    Rloc ++;
    return q [ rloc ];
}
```

Пример шаблона класса

```
int main ( )  {
    Queue < int > a, b;  // две очереди для целых
чисел
    a.qput ( 10 );       b.qput ( 19 );
    a.qput ( 20 );       b.qput (  1 );
    cout << a.qget ( ) << " ";
    cout << a.qget ( ) << " ";
    cout << b.qget ( ) << " ";
    cout << b.qget ( ) << "\n";
    queue < double > c, d; // две очереди для
действительных чисел
    c.qput ( 10.12 );      d.qput ( 19.99 );
    c.qput ( -20.0 );      d.qput (   0.986 );
    cout << c.qget ( ) << " ";
    cout << c.qget ( ) << " ";
    cout << d.qget ( ) << " ";
    cout << d.qget ( ) << "\n";
    return 0;
}
```

Использование параметров, не являющихся типами

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

// Здесь, int size это параметр не являющийся типом.
template < class AType, int size > class atype
{
    AType a [ size ]; // параметр size передает длину массива
public:
    atype ( ) {
     int i;
        for ( i = 0; i < size; i++ ) a [ i ] = i;
    }
    AType & operator[] ( int i );
};

// Обеспечение контроля границ.
template < class AType, int size >
AType & atype < AType, size > :: operator[] ( int i )
{
    if ( i < 0 || i > size - 1 ) {
        cout << "\nЗначение индекса ";
        cout << i << " за пределами границ массива.\n";
        exit ( 1 );
    }
    return a [ i ];
}
```

Использование параметров, не являющихся типами

```
int main ( )
{
    atype < int,    10 > intob;     // массив целых чисел
    atype < double, 15 > doubleob; // массив действительных
                                   //     чисел
    int i;

    cout << "Целочисленный массив: ";
    for ( i = 0; i < 10; i++ ) intob[ i ] = i;
    for ( i = 0; i < 10; i++ ) cout << intob [ i ] << "  ";
    cout << '\n';

    cout << "Действительный массив: ";
    for ( i = 0; i < 15; i++ ) doubleob [ i ] = ( double ) i/3;
    for ( i = 0; i < 15; i++ ) cout << doubleob [ i ] << "  ";
    cout << '\n';

    intob [ 12 ] = 100; // Ошибка времени выполнения

    return 0;
}
```