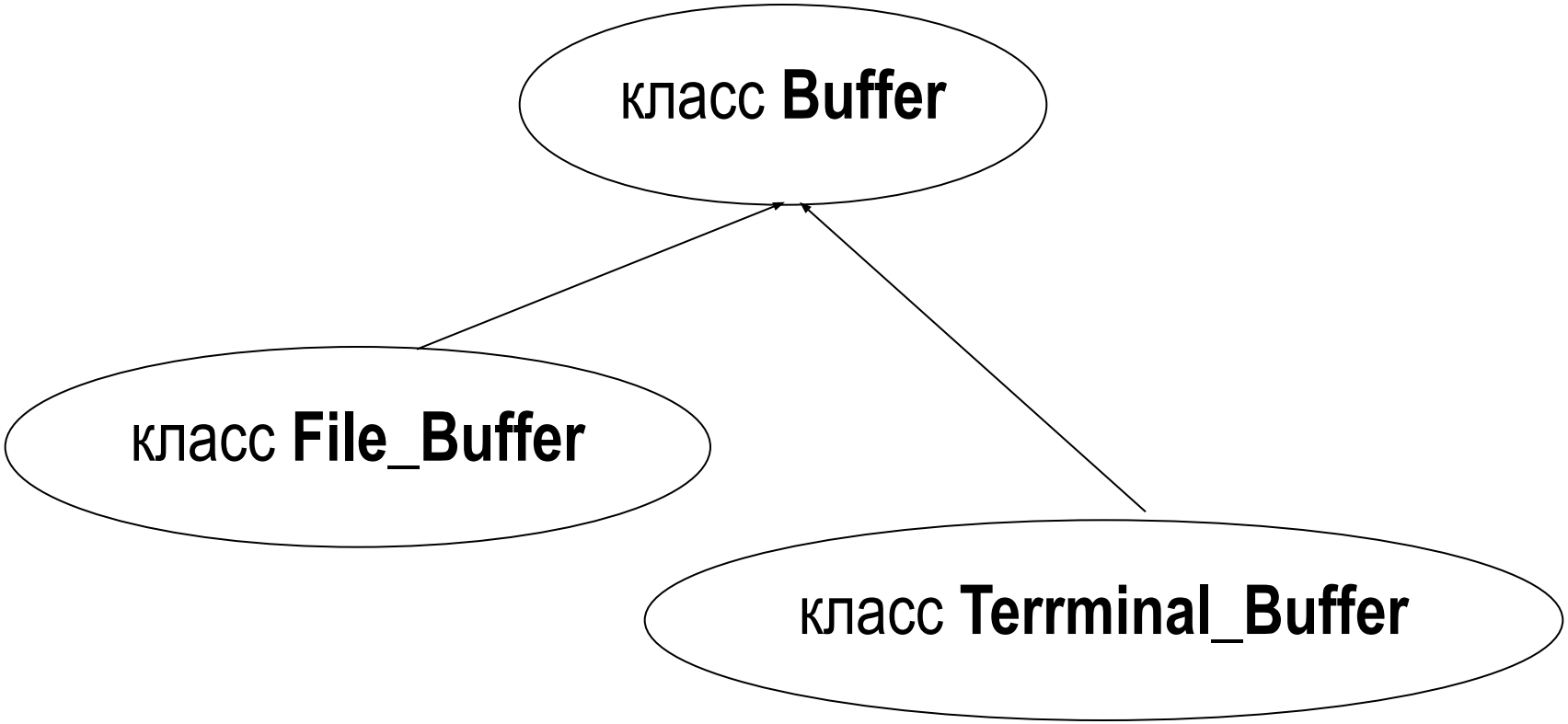
Тема: Доступ в производном классе к базовому классу

- Пример – буферизация вывода данных
- Проблема доступа
- Защищенные члены класса
- Правила доступа

Пример: Класс ***Buffer***

- Реализует понятие буфера данных
- Используется как базовый
- Является абстрактным классом
- Для простоты ограничимся буферизацией при выводе данных

Наследование



Класс ***Buffer***

```
class Buffer {
  public:
    Buffer( int size );
    virtual ~Buffer( );
    void add( char );
  private:
    virtual void flush( ) = 0 ;
     // посылает символы из буфера устройству
    int  _size_limit;
    char* data;
     // указывает на массив из _size_limit  символов
    int  _contains;
    // реально содержится символов в буфере
};
```

Методы класса ***Buffer***

```
Buffer :: Buffer ( int size ) {
    data = new char[size] ;
    if( data == 0 ) {     cerr << “Нет памяти.\n”; exit( ); }
    _size_limit = size;
    _contains = 0;
 }
Buffer :: ~Buffer ( ) { delete data; }
void Buffer ::add( char ch ) {
    if(_contains ==_size_limit ) { flush(); _contains = 0; }
    data[_contains++] = ch;
}
```

Проблема

- ***flush*( ) -** должна брать символы из массива ***data*** и посылать их устройству
- ***flush*( ) -** чистая виртуальная функция и ее реализация будет в производных классах

ПРОБЛЕМА

- функция производного класса не имеет доступа к закрытому свойству базового класса

Решение проблемы

- добавить методы ***get*( )** и ***contains*( )** в класс ***Buffer***
- сделать эти методы защищенными
  - производные классы имеют доступ
  - пользователи не имеют доступа

Класс ***Buffer*** : защищенные методы

```
class Buffer {
  public:
    Buffer( int size );
    virtual ~Buffer( );
    void add( char );
  protected:
    int contains( );
    char get( int index );
  private:
    virtual void flush( ) = 0 ;
    int  _size_limit;
    char* data;
    int  _contains;
};
```

Защищенные методы класса ***Buffer***

```
#include “buffer.h”
int Buffer :: contains (  ) {
    return   _contains;
 }
char Buffer :: get ( int index ) {
     return  data[ index ];
}
```

Класс ***File_Buffer***

```
#include “buffer.h”
#include <fstream.h>
class File_Buffer : public Buffer {
    public:
      File_Buffer( const char* name, int size );
      ~File_Buffer( );
     private:
        void flush( ) ;
        ofstream  str;
};
```

Методы класса ***File_Buffer***

```
#include “file_buffer.h”
File_Buffer :: File_Buffer ( const char* name, int size )
                : Buffer( size ), str( name ) { }
File_Buffer :: ~File_Buffer ( ) {
    if( contains( ) > 0 ) flush( );
}
void File_Buffer :: flush ( ) {
    int i;
    for( i = 0 ; i < contains( ) ; i++ ) str << get(i);
    str.flush();
}
```

Использование буфера

```
#include “file_buffer.h”
#include <fstream.h>
#include <dos.h>
void do_test ( ) {

 File_Buffer  test ( “test_file”, 5 );
 char ch;
 cout << “В начале: \n”;
 system( “dir test_file  > log_file” );
 for( ch=‘a’ ; ch <= ‘e’ ; ch++ )
    test.add(ch);
```

```
 cout << “После 5 символов: \n”;
 system( “dir test_file  > log_file” );
 test.add( ‘f’ );
 cout << “После 6 символов: \n”;
 system( “dir test_file  > log_file” );
}
void main ( void ) {
 do_test ( );
 cout << “После do_test : \n”;
 system( “dir test_file  > log_file” );
}
```

Класс ***Terminal_Buffer***

```
#include “buffer.h”
class Terminal_Buffer : public Buffer {
    public:
      Terminal_Buffer( int size );
      ~Terminal_Buffer( );
     private:
       void flush( ) ;
};
```

Методы класса ***Terminal_Buffer***

```
#include “term_buffer.h”
Terminal_Buffer :: Terminal_Buffer ( int size )
                : Buffer( size ) { }
Terminal_Buffer :: ~Terminal_Buffer ( )  {
    if( contains( ) >  0 ) flush( );
}
void Terminal_Buffer :: flush ( ) {
    int i;
    for( i = 0 ; i < contains( ) ; i++ )  cerr << get(i);
}
```

Наследование

Базовый класс	Производный		
	public	private	protected
public	public	private	protected
private	недоступно	недоступно	недоступно
protected	private	private	private

РЕЗЮМЕ

- Пользователи класса имеют доступ только к открытым членам класса
- Производный класс имеет доступ к открытым и защищенным членам класса
- При использовании класса надо знать его открытые члены и функции-друзья
- При создании производного класса надо знать также
  - защищенные члены класса
  - закрытые виртуальные функции класса