Лекция 2: архитектура приложения

- **Функциональные требования:**
  - постановка задачи;
  - сбор требований;
  - анализ требований.
- **Описание системы:**
  - определение рабочих ситуаций;
  - список сценариев;
  - детализация рабочих ситуаций посредством описания потока событий.
- **Разработка архитектуры:**
  - определение архитектурного стиля;
  - выбор парадигмы программирования;
  - выбор стандартных технологий и их конкретных реализаций.
- **Логическая модель:**
  - создание паттернов разработки;
  - определение классов и их взаимосвязей;
  -

Функциональные требования

Задание:

Разработать калькулятор для работы с матрицами большой размерности над произвольным полем. Необходимо реализовать стандартные операции – транспонирование, умножение матрицы на число, сложение матриц, умножение матриц, нахождение обратной матрицы, и предусмотреть реализацию произвольных линейных и нелинейных операторов.

Требования:

Получение результата запроса – вычисления выражения с матричными операндами, должно происходить в режиме реального времени.

Пользователь может работать с калькулятором на различных вычислительных устройствах – персональном компьютере, планшете или мобильном телефоне.

Расширение возможностей калькулятора не должно сопровождаться обновлением программного обеспечения пользователя.

Рабочая ситуация

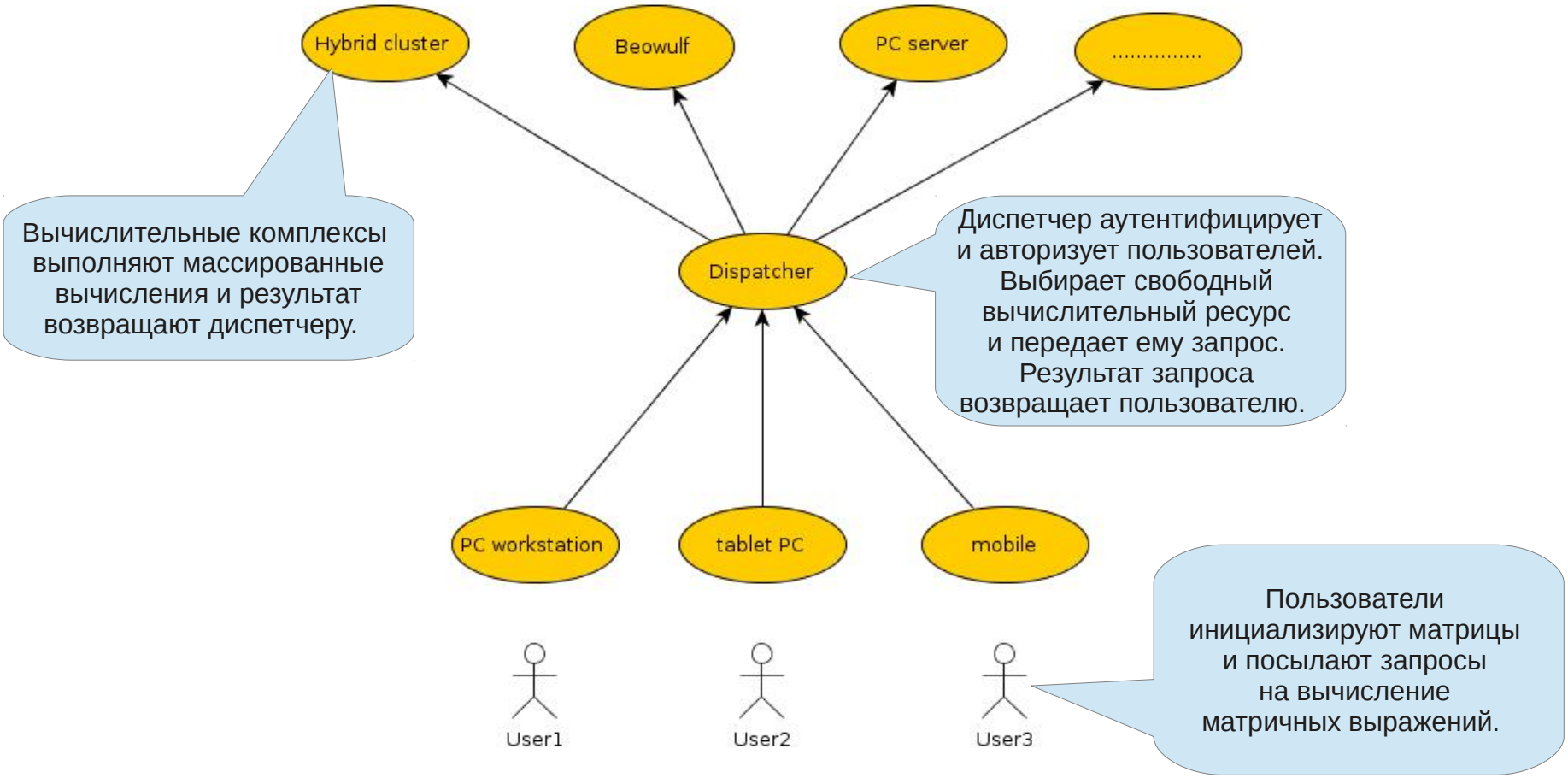
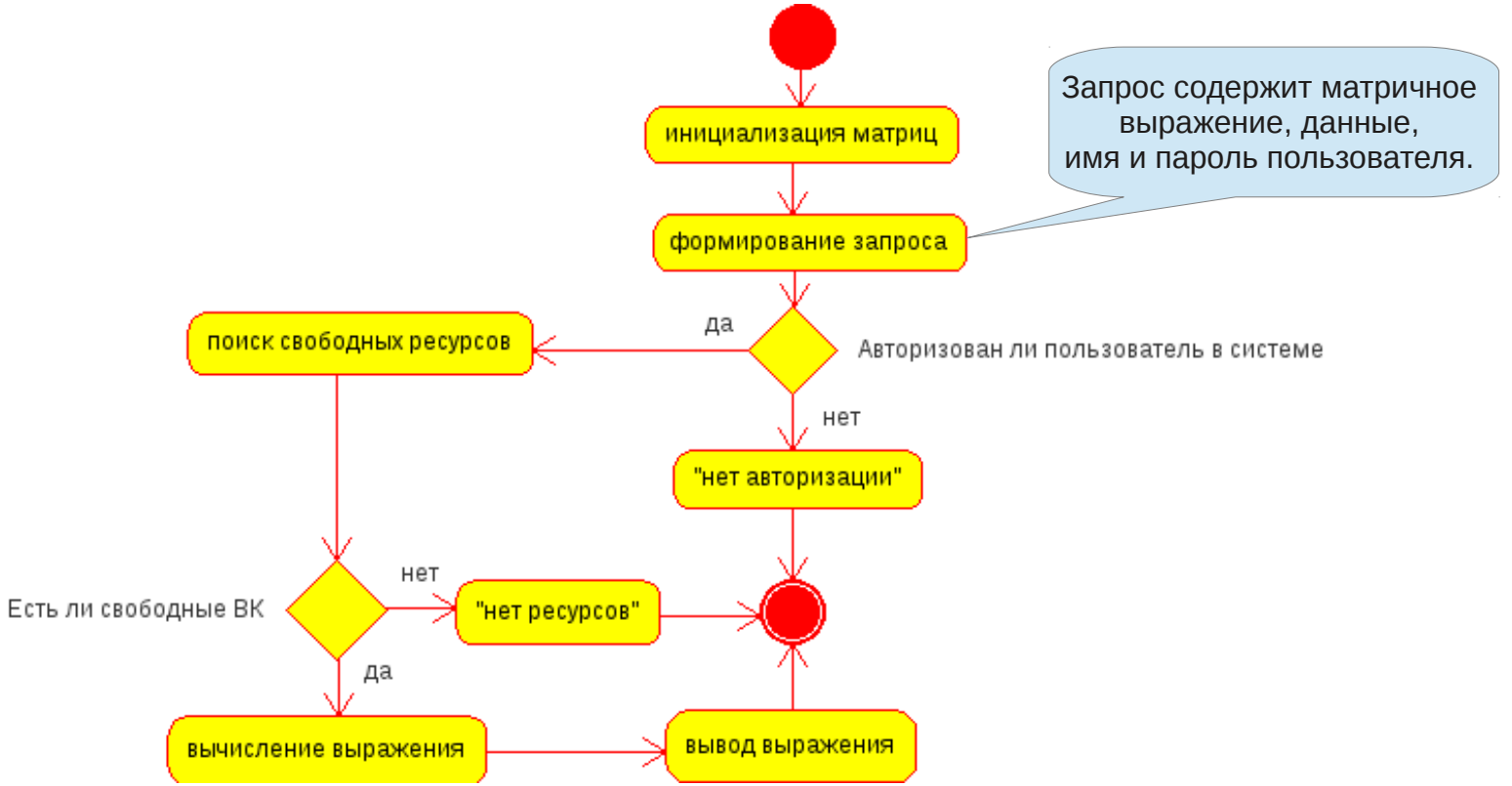
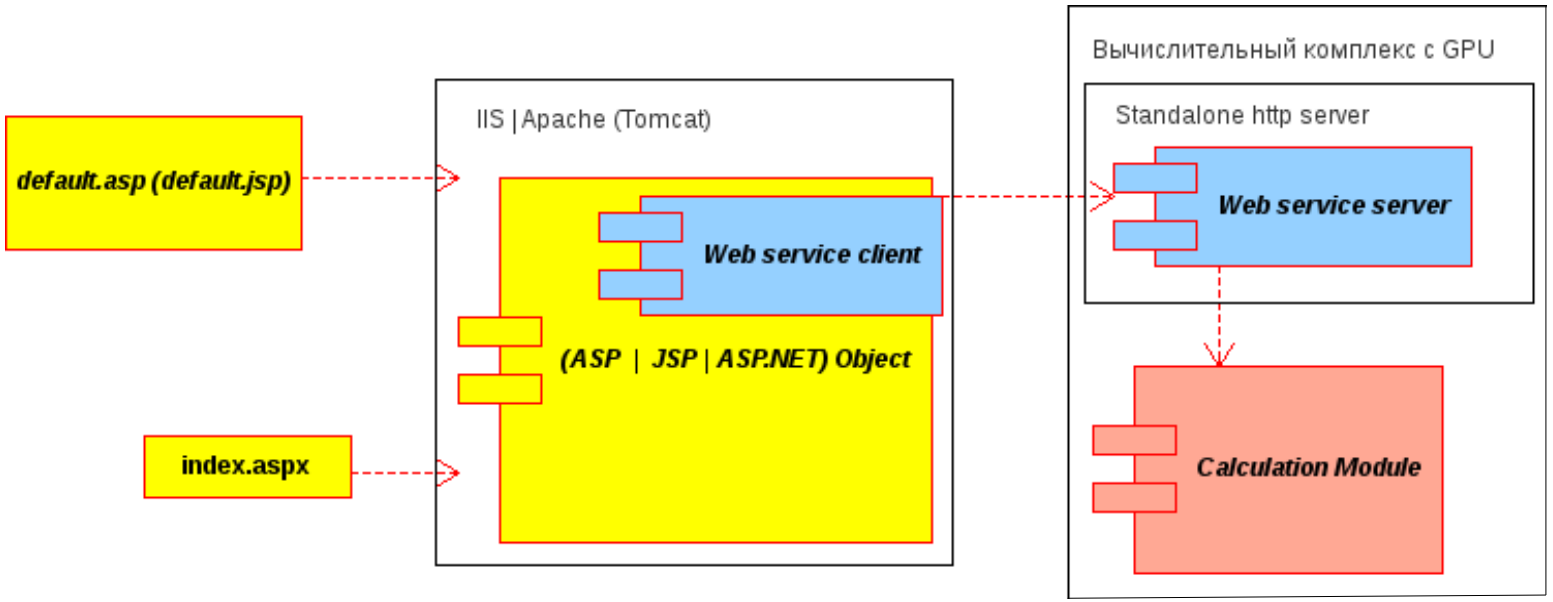


Диаграмма активности



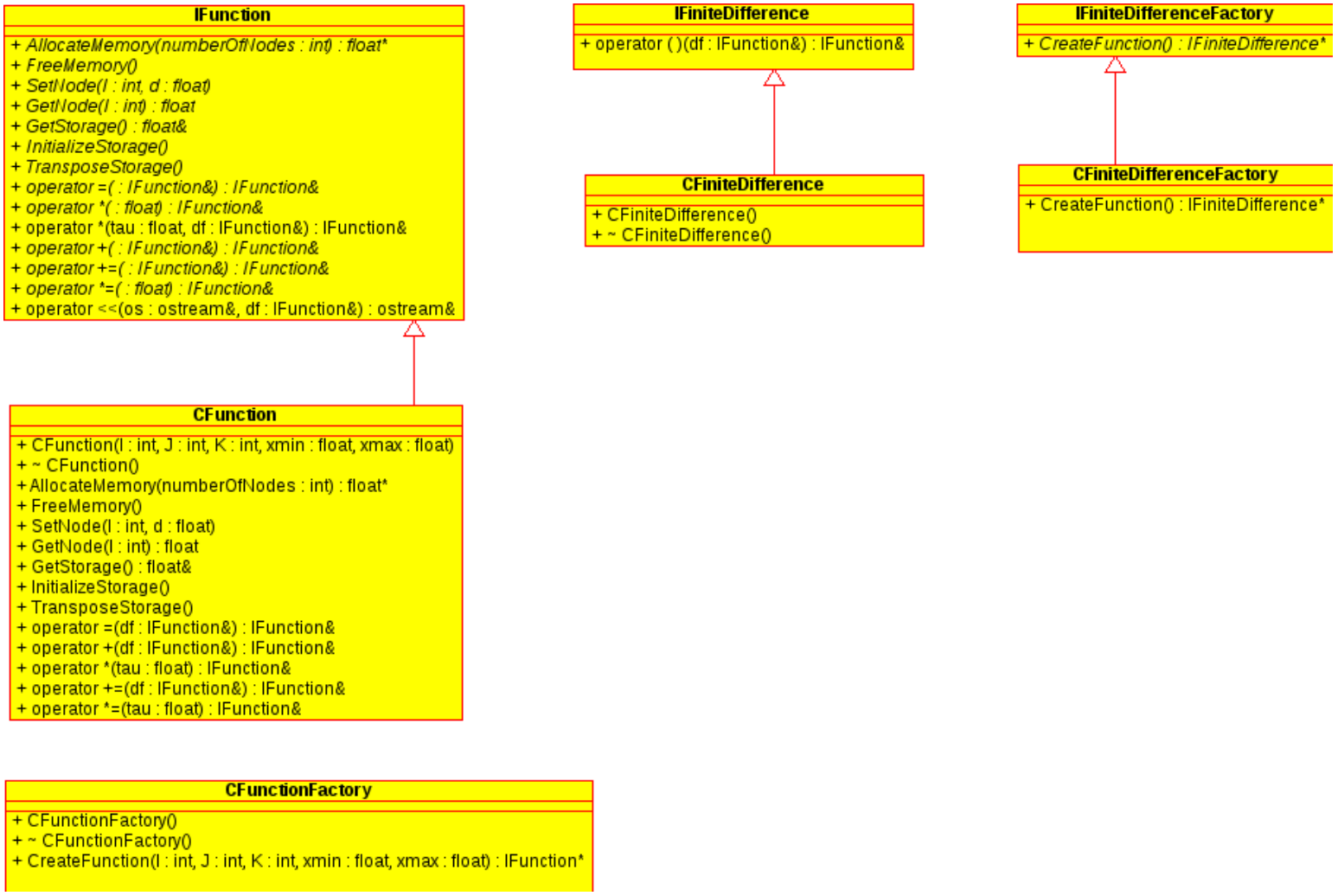
Основные компоненты приложения и их распределение



Этапы разработки

	Платформа	Разделы курсов	Дата выполнения
“Вычислитель”	Linux, CUDA, C/C++, nvcc, gcc (g++)	Программирование CUDA C	26.10.2013
“Диспетчер”	Linux | Windows, Web services, gSOAP	COM – модель составных компонентов, DCOM – распределенные приложения на основе COM, ***Web- services***	16.11.2013
“Клиент”	Linux | Windows, asp | asp.net | jsp | php, c#, java script, xml+xslt+css		30.11.2013
“Развертывание”	Linux | Windows, CMake	Cmake [,Autotools]	21.12.2013

Вычислительная компонента (*макет*)



Программная реализация вычислительной компоненты: интерфейс IFunction

```
/*
 * IFunction.h
 *
 *  Created on: Sep 25, 2013
 *      Author: ewgenij
 */

#ifndef IFUNCTION_H_
#define IFUNCTION_H_
#include <iostream>
using namespace std;

class IFunction {
public:
        //IFunction();
        //virtual ~IFunction();
        virtual float* AllocateMemory(int numberOfNodes)=0;
        virtual void FreeMemory()=0;
        virtual void SetNode(int I, float d)=0;
        virtual float GetNode(int I)=0;
        virtual float& GetStorage()=0;
        virtual void InitializeStorage()=0;
        virtual void TransposeStorage()=0;
```

Программная реализация вычислительной компоненты: интерфейс IFunction (продолжение)

```
        virtual IFunction& operator=(IFunction&)=0;
        virtual IFunction& operator*(float)=0;
        friend IFunction& operator*(float tau, IFunction& df){
                return df*tau;
        }
        virtual IFunction& operator+(IFunction&)=0;
        virtual IFunction& operator+=(IFunction&)=0;
        virtual IFunction& operator*=(float)=0;


        friend ostream& operator<<(ostream& os, IFunction& df){
                df.OutputStorage(os);
                return os;
        }
private:
        virtual void OutputStorage(ostream& os)=0;
};

#endif /* IFUNCTION_H_ */
```

Программная реализация вычислительной компоненты: интерфейс IDifference

```
/*
 * IFiniteDifference.h
  *  Created on: Sep 25, 2013
 *      Author: ewgenij
 */

#ifndef IFINITEDIFFERENCE_H_
#define IFINITEDIFFERENCE_H_

#include "IFunction.h"

class IFiniteDifference {
public:
	//IFiniteDifference();
	//virtual ~IFiniteDifference();
	IFunction& operator()(IFunction& df){
		df=CalcFiniteDifference(df);
		return df;
	}
private:
	virtual IFunction& CalcFiniteDifference(IFunction& df)=0;
};
#endif /* IFINITEDIFFERENCE_H_ */
```

Программная реализация вычислительной компоненты: реализация CFunction

```
/*
 * CFunction.h
 *
 *  Created on: Sep 25, 2013
 *      Author: ewgenij
 */

#ifndef CFUNCTION_H_
#define CFUNCTION_H_

#include "IFunction.h"
#include <cmath>
using namespace std;

class CFunction : public IFunction {
public:
        CFunction(int I, int J, int K, float xmin, float xmax);
        virtual ~CFunction(){
                                FreeMemory();
                        };
```

Программная реализация вычислительной компоненты: реализация CFunction (*продолжение*)

```
float* AllocateMemory(int numberOfNodes){
        N=numberOfNodes;
        storage=new float[numberOfNodes*sizeof(float)];
        return storage;
}
virtual void FreeMemory(){
        delete storage;
}

void SetNode(int I,float d){storage[I]=d;}
float GetNode(int I){return storage[I];}

float& GetStorage(){
        return *storage;
}

void InitializeStorage();
void TransposeStorage();
```

Программная реализация вычислительной компоненты: реализация CFunction (*продолжение*)

```
virtual IFunction& operator=(IFunction& df){
		for(int i=0;i<N;i++){
			storage[i]=df.GetNode(i);
		}
		return *this;
	}

	virtual IFunction& operator+(IFunction& df){
		for(int i=0;i<N;i++){
			storage[i]+=df.GetNode(i);
		}
		return *this;
	}

	virtual IFunction& operator*(float tau){
		for(int i=0;i<N;i++){
			storage[i]*=tau;
		}
		return *this;
	}
```

Программная реализация вычислительной компоненты: реализация CFunction (*продолжение*)

```
        IFunction& operator+=(IFunction& df){
                for(int i=0;i<N;i++){
                        storage[i]+=df.GetNode(i);
                }
                return *this;
        }

        IFunction& operator*=(float tau){
                for(int i=0;i<N;i++){
                        storage[i]*=tau;
                }
                return *this;
        }
private:
        float* storage;

        void OutputStorage(ostream& os){
                 for(int i=0;i<I; i++)
                        //for(int j=0;j<J; j++)
                          for(int k=0;k<K; k++)
                                //cout<<i<<'\t'<<j<<'\t'<<k<<'\t'<<storage[k+j*K+i*K*J]<<endl;
                                  cout<<i<<'\t'<<k<<'\t'<<storage[k+(J/2)*K+i*K*J]<<endl;
        }
```

Программная реализация вычислительной компоненты: реализация CFunction (*продолжение*)

```
        int N;
        int I,J,K,L;
        float xmin, xmax;

inline float Normal(float x,float y, float z){
                const float PI=3.14159265358979323846;
                const float T=0.1f;
                const float x0=(xmax+xmin)/2.0+0.1*(xmax-xmin);
                return
                exp( -( (x-x0)*(x-x0)+y*y+z*z )/2.0/T)/pow(2.0*PI*T, 1.5);
        }
};

#endif /* CFUNCTION_H_ */
```

Программная реализация вычислительной компоненты: реализация CFunction (*продолжение*)

```
/*
 * CFunction.cpp
 *  Created on: Sep 25, 2013
 *      Author: ewgenij
 */
#include "CFunction.h"
    CFunction::CFunction(int I, int J, int K, float xmin, float xmax):
                          I(I),  J(J),  K(K),    xmin(xmin), xmax(xmax){
          AllocateMemory(I*J*K);
          InitializeStorage();
    }
void CFunction::InitializeStorage(){
          int i,j,k;
          float h=(xmax-xmin)/I;
          float t;
          float x, y, z;
    for(i=0, x=xmin;i<I; x+=h, i++)
           for(j=0, y=xmin; j<J; y+=h, j++)
            for(k=0, z=xmin; k<K; z+=h, k++)
                  storage[k+j*K+i*K*J]=Normal(x,y,z);
}
void CFunction::TransposeStorage(){}
```

Программная реализация вычислительной компоненты: реализация CFiniteDifference

```
/*
 * CFiniteDifference.h
 *
 *  Created on: Sep 25, 2013
 *      Author: ewgenij
 */

#ifndef CFINITEDIFFERENCE_H_
#define CFINITEDIFFERENCE_H_

#include "IFiniteDifference.h"

class CFiniteDifference : public IFiniteDifference {
public:
	CFiniteDifference();
	virtual ~CFiniteDifference();
private:
	virtual IFunction& CalcFiniteDifference(IFunction& df){
		return 2.0*df;
	}
};

#endif /* CFINITEDIFFERENCE_H_ */
```

Программная реализация вычислительной компоненты: реализация CFiniteDifference

```
/*
 * CFiniteDifference.h
 *
 *  Created on: Sep 25, 2013
 *      Author: ewgenij
 */

#ifndef CFINITEDIFFERENCE_H_
#define CFINITEDIFFERENCE_H_

#include "IFiniteDifference.h"

class CFiniteDifference : public IFiniteDifference {
public:
	CfiniteDifference();
	virtual ~CfiniteDifference();
private:
	virtual IFunction& CalcFiniteDifference(IFunction& df){
		return 2.0*df;
	}
};

#endif /* CFINITEDIFFERENCE_H_ */
```

Программная реализация вычислительной компоненты: реализация CFiniteDifference (*продолжение*)

```
/*
 * CFiniteDifference.cpp
 *
 *  Created on: Sep 25, 2013
 *      Author: ewgenij
 */

#include "CFiniteDifference.h"

CFiniteDifference::CFiniteDifference()
{
	// TODO Auto-generated constructor stub

}

CFiniteDifference::~CFiniteDifference()
{
	// TODO Auto-generated destructor stub
}
```

Фабрика классов: интерфейс IFiniteDifferenceFactory

```
/*
 * IFiniteDifferenceFactory.h
 *
 *  Created on: Sep 25, 2013
 *      Author: ewgenij
 */

#ifndef IFINITEDIFFERENCEFACTORY_H_
#define IFINITEDIFFERENCEFACTORY_H_
#include "IFiniteDifference.h"

class IFiniteDifferenceFactory {
public:
        virtual IFiniteDifference* CreateFunction()=0;
};

#endif /* IFINITEDIFFERENCEFACTORY_H_ */
```

Фабрика классов: реализация CFiniteDifferenceFactory

```
/*
 * CFiniteDifferenceFactory.h
 *
 *  Created on: Sep 25, 2013
 *      Author: ewgenij
 */

#ifndef CFINITEDIFFERENCEFACTORY_H_
#define CFINITEDIFFERENCEFACTORY_H_
#include "IFiniteDifferenceFactory.h"

class CFiniteDifferenceFactory : public IFiniteDifferenceFactory{
public:
        virtual IFiniteDifference* CreateFunction();
};

#endif /* CFINITEDIFFERENCEFACTORY_H_ */
```

Фабрика классов: реализация CFiniteDifferenceFactory (*продолжение*)

```
/*
 * CFiniteDifferenceFactory.cpp
 *
 *  Created on: Sep 25, 2013
 *      Author: ewgenij
 */

#include "CFiniteDifferenceFactory.h"
#include "CFiniteDifference.h"

IFiniteDifference* CFiniteDifferenceFactory::CreateFunction(){
        return new CFiniteDifference;
}
```

Фабрика классов: реализация CFunctionFactory

```
/*
 * CFunctionFactory.h
 *
 *  Created on: Sep 25, 2013
 *      Author: ewgenij
 */

#ifndef CFUNCTIONFACTORY_H_
#define CFUNCTIONFACTORY_H_
#include "IFunction.h"

class CFunctionFactory {
public:
        CFunctionFactory();
        virtual ~CFunctionFactory();
        virtual IFunction* CreateFunction(int I, int J, int K, float xmin, float xmax);
};

#endif /* CFUNCTIONFACTORY_H_ */
```

Фабрика классов: реализация CFunctionFactory (*продолжение*)

```
/*
 * CFunctionFactory.cpp
 *
 *  Created on: Sep 25, 2013
 *      Author: ewgenij
 */
#include "CFunctionFactory.h"
#include "CFunction.h"

CFunctionFactory::CFunctionFactory()
{
	// TODO Auto-generated constructor stub

}

CFunctionFactory::~CFunctionFactory()
{
	// TODO Auto-generated destructor stub
}

IFunction* CFunctionFactory::CreateFunction(int I, int J, int K, float xmin, float xmax){
	return new CFunction(I, J, K, xmin, xmax);
}
```

Пример использования

```
/*
 * main.cpp
 *  Created on: Sep 25, 2013
 *      Author: ewgenij
 */
#include "IFunction.h"
#include "IFiniteDifference.h"
#include "CFunctionFactory.h"
#include "CFiniteDifferenceFactory.h"
int main(){
        int I,J,K;
        float xmin=-1.0, xmax=1.0;
        I=J=K=32;
        IFunction* pdf;
        CFunctionFactory dfF;
        pdf=dfF.CreateFunction(I,J,K,xmin,xmax);
        IFunction& df=*pdf;

        IFiniteDifference* pD;
        CFiniteDifferenceFactory fdF;
        pD=fdF.CreateFunction();
        IFiniteDifference& D=*pD;

        float tau=0.01;     df=tau*D(df);  cout<<df;  return 0;  }
```

Задания

- реализовать интерфейс IFunction на основе последовательного алгоритма и на основе CUDA;
- реализовать интерфейс IfiniteDifference на основе последовательного алгоритма и на основе CUDA.