МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«КАЗАНСКИЙ (ПРИВОЛЖСКИЙ) ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

П Р И К А З

26 марта 2019 г. **№ 01/932**

В соответствии с утвержденным учебным планом направления 10.03.01 «Информационная безопасность» профиль «Безопасность автоматизированных систем» и приказа № 01-03/6 от 09.01.2019 «Об организации образовательного процесса во втором семестре 2018/2019 учебного года в связи с проведением 45 мирового чемпионата по профессиональному мастерству по стандартам World Skills» направить для прохождения учебной практики обучающихся 1 курса 06- 851 гр. с 01 июля 2019 г. по 14 июля 2019 г. с назначением руководителей практики

ФИО студента	Группа	Место прохождения практики	ФИО руководителя от кафедры
1. Ажалиев Райнур Руфатович	06-852	Кафедра радиофизики, КФУ	Скворцов И. В.
2. Алейникова Александра Антоновна	06-852	Кафедра радиофизики, КФУ	Скворцов И. В.
3. Алексеев Андрей Эдуардович	06-852	Кафедра радиофизики, КФУ	Скворцов И. В.
4. Быстрин Матвей Сергеевич	06-852	Кафедра радиофизики, КФУ	Скворцов И. В.
5. Вартикян Люсине Гамлетовна	06-852	Кафедра радиофизики, КФУ	Скворцов И. В.
6. Габдрахимов Рамиль Ринатович	06-852	Кафедра радиофизики, КФУ	Скворцов И. В.
7. Галимзанова Рената Марселевна	06-852	Кафедра радиофизики, КФУ	Скворцов И. В.
8. Гильмутдинова Регина	06-852	Кафедра радиофизики, КФУ	Скворцов И. В.

В соответствии с утвержденным учебным планом Института физики направления 10.03.01 «Информационная безопасность» профиль «Обеспечение информационной безопасности распределенных информационных систем» и приказа № 01-03/6 от 09.01.2019 «Об организации образовательного процесса во втором семестре 2018/2019 учебного года в связи с проведением 45 мирового чемпионата по профессиональному мастерству по стандартам World Skills» направить для прохождения учебной практики (Ознакомительная практика) обучающихся 1 курса 06- 852 гр. с 01 июля 2019 г. по 14 июля 2019 г. с назначением руководителей практики

ФИО студента	Группа	Место прохождения практики	ФИО руководителя от кафедры
1. Ахметова Диана Ильдаровна	06-852	Кафедра радиофизики, КФУ	Шерстюков О. Н.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
"Казанский (Приволжский) федеральный университет"
Институт физики

УТВЕРЖДАЮ
Проректор по образовательной деятельности КФУ
проф. Таюрский Д.А.
"___"_____20___ г.

Программа дисциплины

Языки программирования Б1.В.06

Направление подготовки: 10.03.01 - Информационная безопасность
Профиль подготовки: Безопасность автоматизированных систем
Квалификация выпускника: бакалавр
Форма обучения: очное
Язык обучения: русский
Год начала обучения по образовательной программе: 2018
Автор(ы): Корчагин П.А.
Рецензент(ы): Шерстюков О.Н.

Шифр компетенции	Расшифровка приобретаемой компетенции
ОПК-4	способностью понимать значение информации в развитии современного общества, применять информационные технологии для поиска и обработки информации
ПК-2	способностью применять программные средства системного, прикладного и специального назначения, инструментальные средства, языки и системы программирования для решения профессиональных задач
ПК-3	способностью администрировать подсистемы информационной безопасности объекта защиты

Форма контроля	Процедура оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций	Этап	Количество баллов
Семестр 2			
Текущий контроль			
Контрольная работа	Контрольная работа проводится в часы аудиторной работы. Обучающиеся получают задания для проверки усвоения пройденного материала. Работа выполняется в письменном виде и сдаётся преподавателю. Оцениваются владение материалом по теме работы, аналитические способности, владение методами, умения и навыки, необходимые для выполнения заданий.	1	10
Лабораторные работы	В аудитории, оснащённой соответствующим оборудованием, обучающиеся проводят учебные эксперименты и тренируются в применении практико-ориентированных технологий. Оцениваются знание материала и умение применять его на практике, умения и навыки по работе с оборудованием в соответствующей предметной области.	2 3	20 20
Экзамен	Экзамен нацелен на комплексную проверку освоения дисциплины. Экзамен проводится в устной или письменной форме по билетам, в которых содержатся вопросы (задания) по всем темам курса. Обучающемуся даётся время на подготовку. Оценивается владение материалом, его системное освоение, способность применять нужные знания, навыки и умения при анализе проблемных ситуаций и решении практических заданий.		50

7. Перечень основной и дополнительной учебной литературы, необходимой для освоения дисциплины (модуля)

7.1 Основная литература:

1. Программирование на языке высокого уровня. Программирование на языке С++: учеб. пособие / Т.И. Немцова, С.Ю. Голова, А.И. Терентьев ; под ред. Л.Г. Гагариной. ? М. : ИД 'ФОРУМ' : ИНФРА-М, 2018. ? 512 с. ? (Среднее профессиональное образование). - Режим доступа: http://znanium.com/catalog/product/918098

2. Немцова Т. И. Программирование на языке высокого уровня. Программир. на языке С++: Уч. пос. / Т.И.Немцова и др.; Под ред. Л.Г.Гагариной - М.: ИД ФОРУМ: ИНФРА-М, 2012. - 512 с. http://znanium.com/bookread.php?book=244875

3. Солонина, А. И. Цифровая обработка сигналов. Моделирование в MATLAB / А. И. Солонина, С. М. Арбузов. - СПб.: БХВ-Петербург, 2008.- 814 с.: ил. - (Учебное пособие) - ISBN 978-5-9775-0259-7.Режим доступа:http://znanium.com/bookread.php?book=350520

7.2. Дополнительная литература:

1. Измерения в LabVIEW/БаранЕ.Д., МорозовЮ.В. - Новосиб.: НГТУ, 2010. - 162 с.: ISBN 978-5-7782-1428-6 - Режим доступа: http://znanium.com/catalog/product/546030

2. Анализ и обработка сигналов в среде MATLAB/ЩетининЮ.И. - Новосиб.: НГТУ, 2011. - 115 с.: ISBN 978-5-7782-1807-9 - Режим доступа: http://znanium.com/catalog/product/548133

Выпускник, освоивший дисциплину:

Должен знать:

Основные функциональные возможности современных универсальных языков программирования высокого уровня

Должен уметь:

Применять полученные знания для решения задач связанных с разработкой программного обеспечения

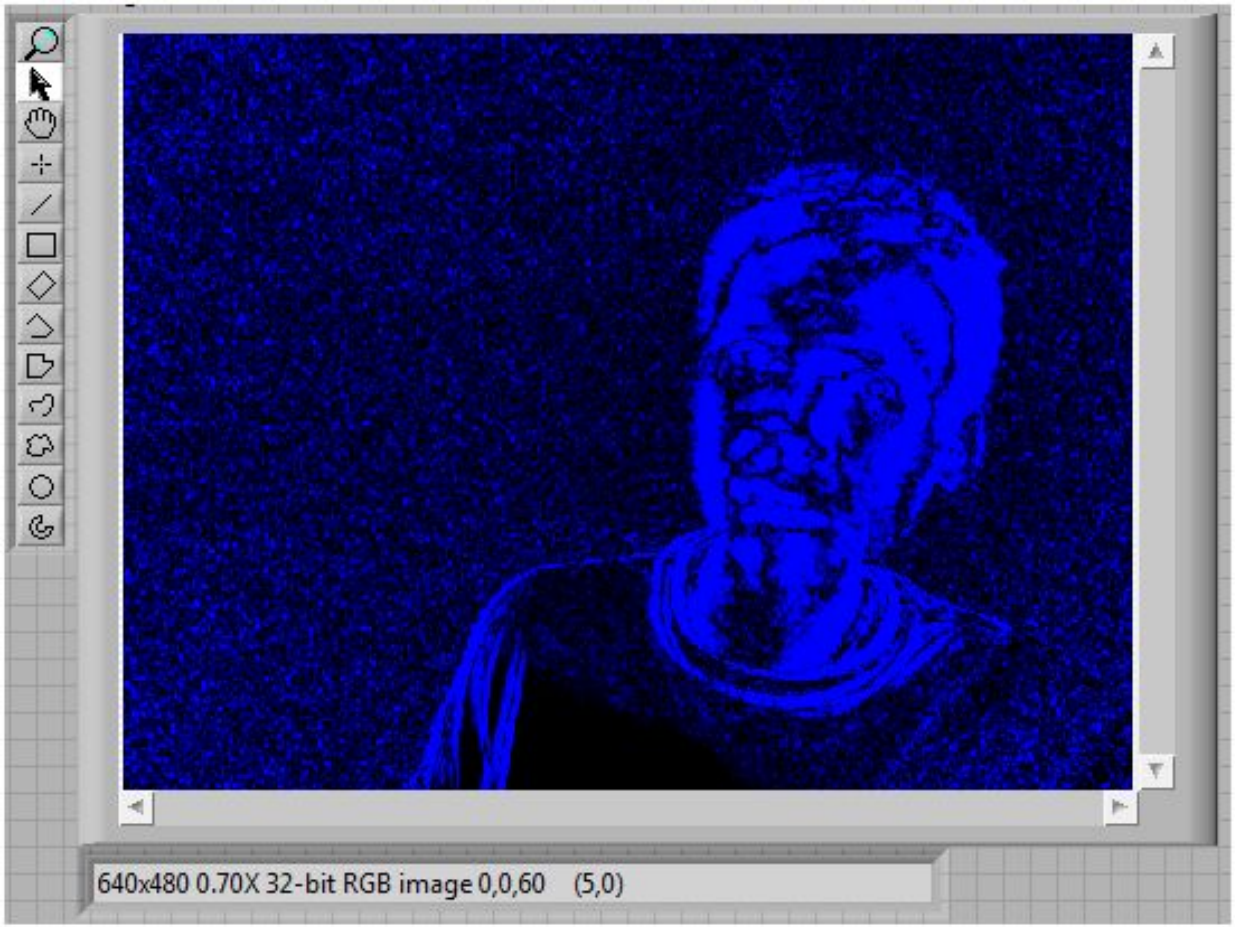
Должен владеть:

Владеть навыками разработки программ в различных средах программирования

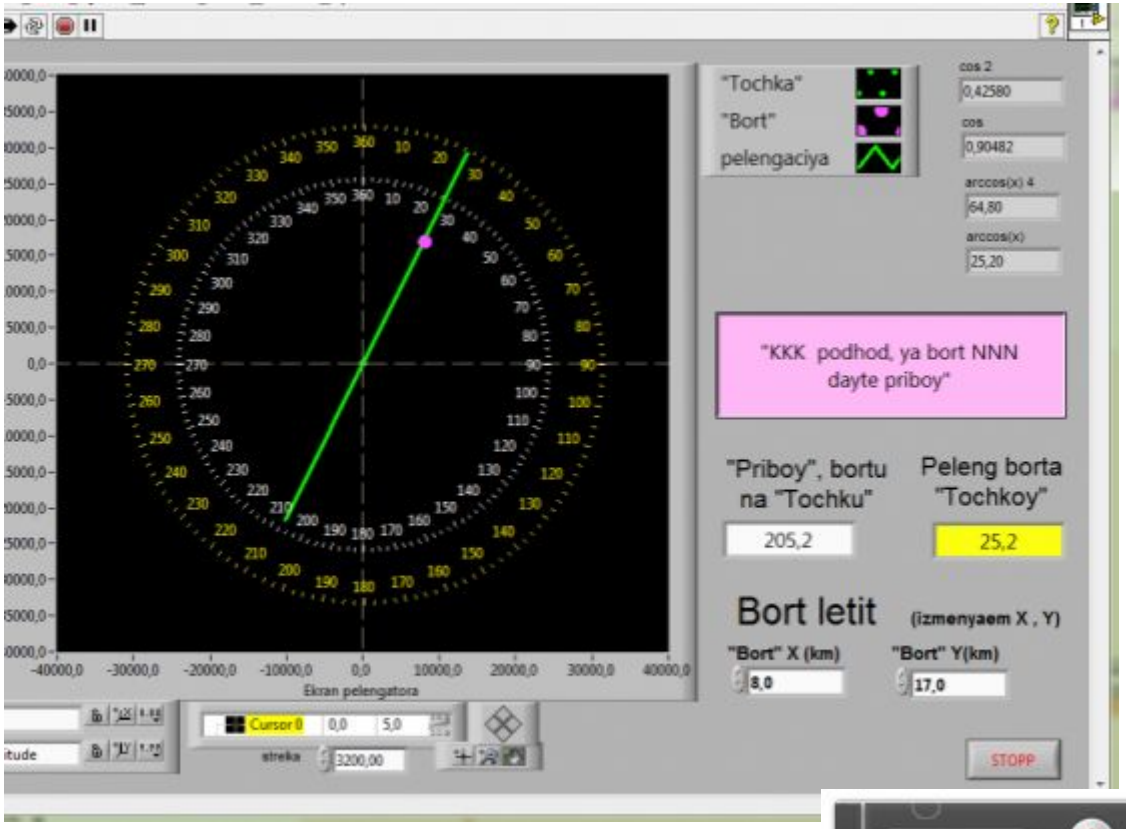
Должен демонстрировать способность и готовность:

Решать прикладные задачи с использованием современных сред и языков программирования

Примеры приборов



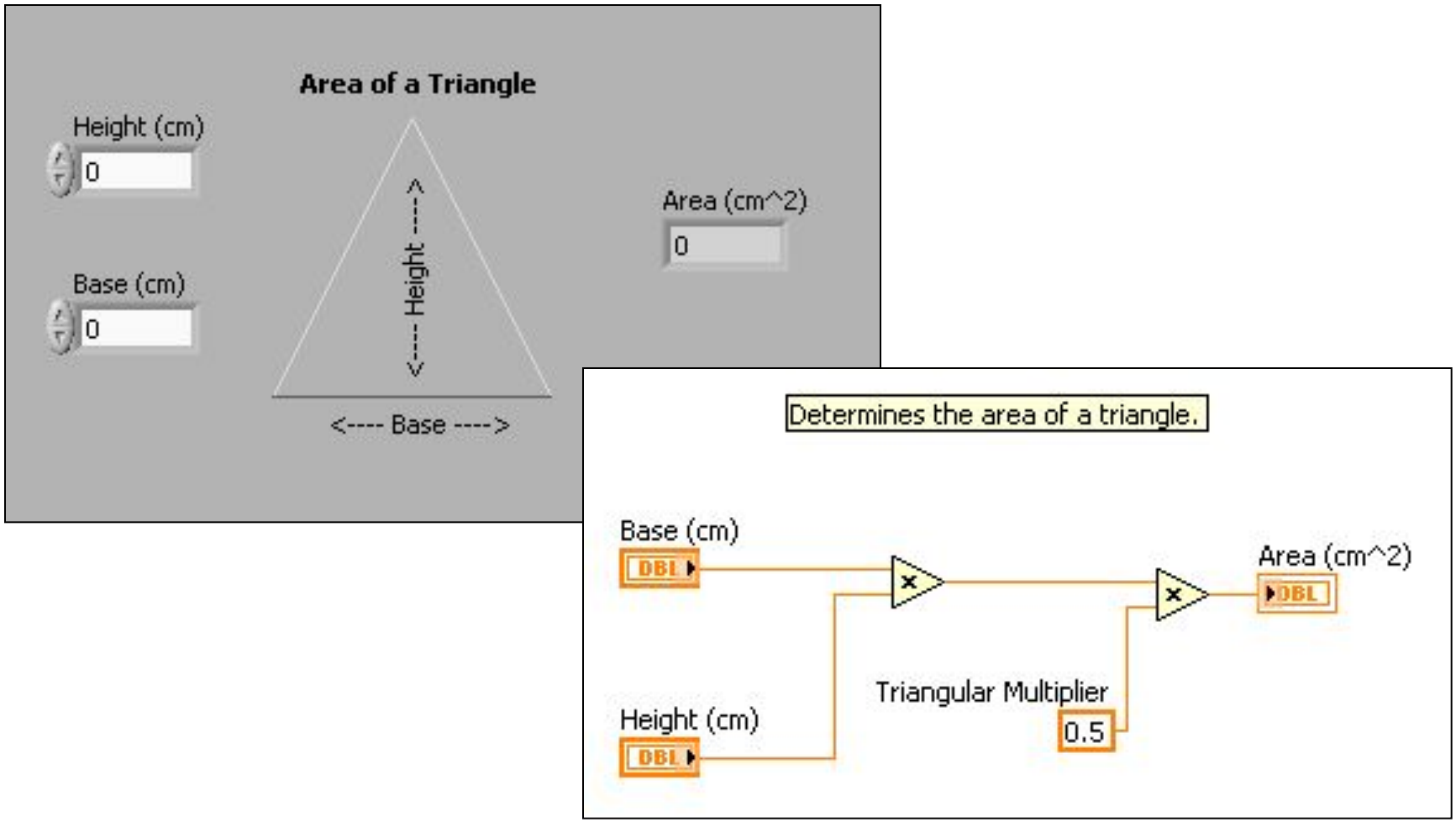
Примеры приборов



F. Block Diagram – Терминалы

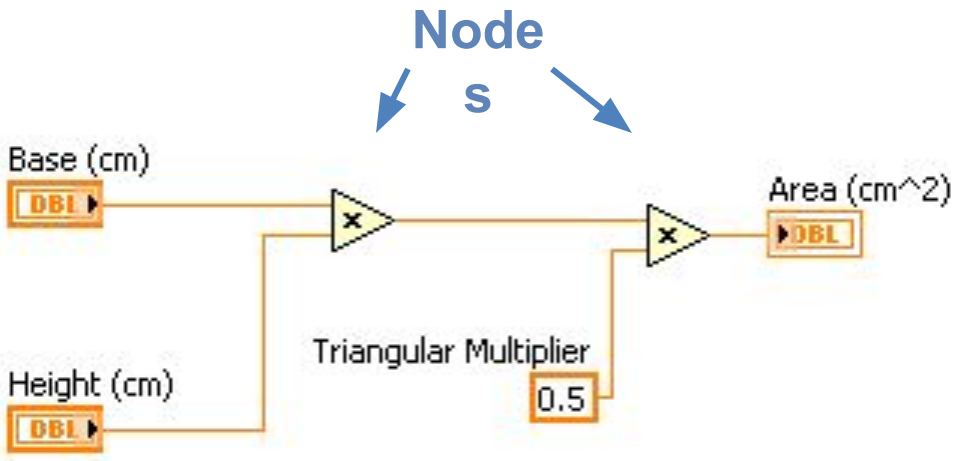
- Терминалы - это:
  - Представление объектов лицевой панели на блок-диаграмме
  - Порты ввода и вывода, через которые осуществляется обмен информацией между лицевой панелью и блок-диаграммой
  - Аналоги параметров и констант в текстовых языках программирования
- Внешний вид терминалов можно изменить, выбрав и переключив пункт **View as Icon** контекстного менк

F. Терминалы блок-диаграммы



F. Block Diagram – Узлы

- Объекты блок-диаграммы, у которых есть входы и/или выходы, и которые выполняют операции при запуске VI
- Аналоги высказываний, операторов, функций и подпрограмм в текстовых языках программирования
- Узлами могут быть функции, subVI или структуры

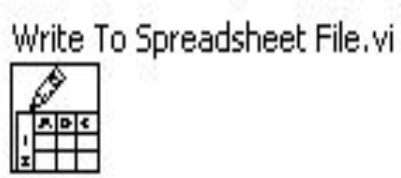


F. Block Diagram – Узлы функций

- Базовые операционные элементы LabVIEW
- Не имеют лицевой панели или блок-диаграммы, но имеют панель подключения
- Двойной щелчок только выделяет функцию, но не раскрывает ее, как в VI
- Фон иконки – бледно-желтый

F. Block Diagram – Узлы SubVI

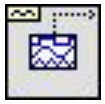
- SubVI – это VI, которые создаются для использования внутри других VI
- Любой VI потенциально может быть использован в качестве subVI
- Если щелкнуть дважды по subVI на блок-диаграмме, то можно увидеть лицевую панель и блок-диаграмму subVI
  - В верхнем правом углу лицевой панели находится иконка текущего VI
  - Эта иконка и появляется на блок-диаграмме, когда VI помещается на блок-диаграмму в качестве subVI

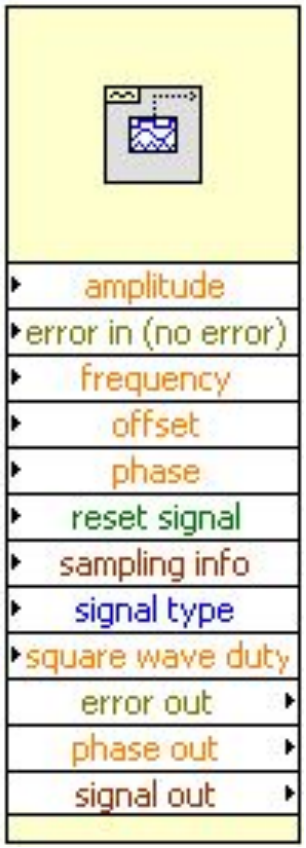


F. Block Diagram – Узлы SubVI

- Express VI – специальный тип subVI
  - Требуют минимума соединений, поскольку их конфигурируют с помощью диалоговых окон
  - Конфигурацию Express VI можно сохранить, как subVI
- Иконки Express VI на блок-диаграмме окружены голубым полем

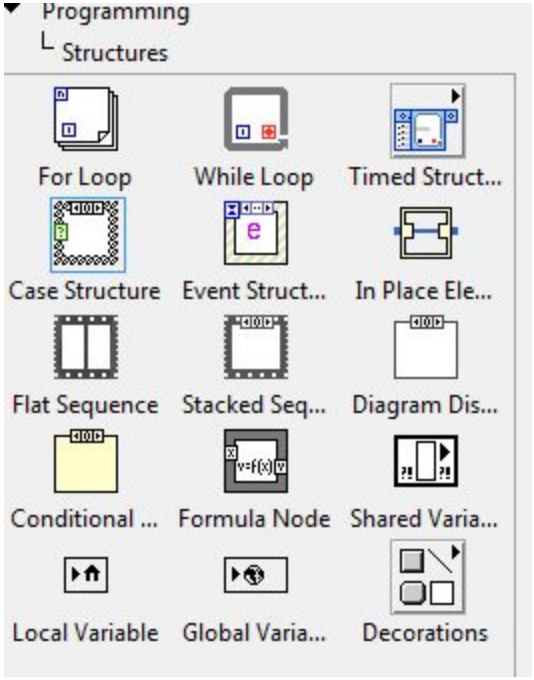
F. Block Diagram – Иконки/Расширяемые узлы





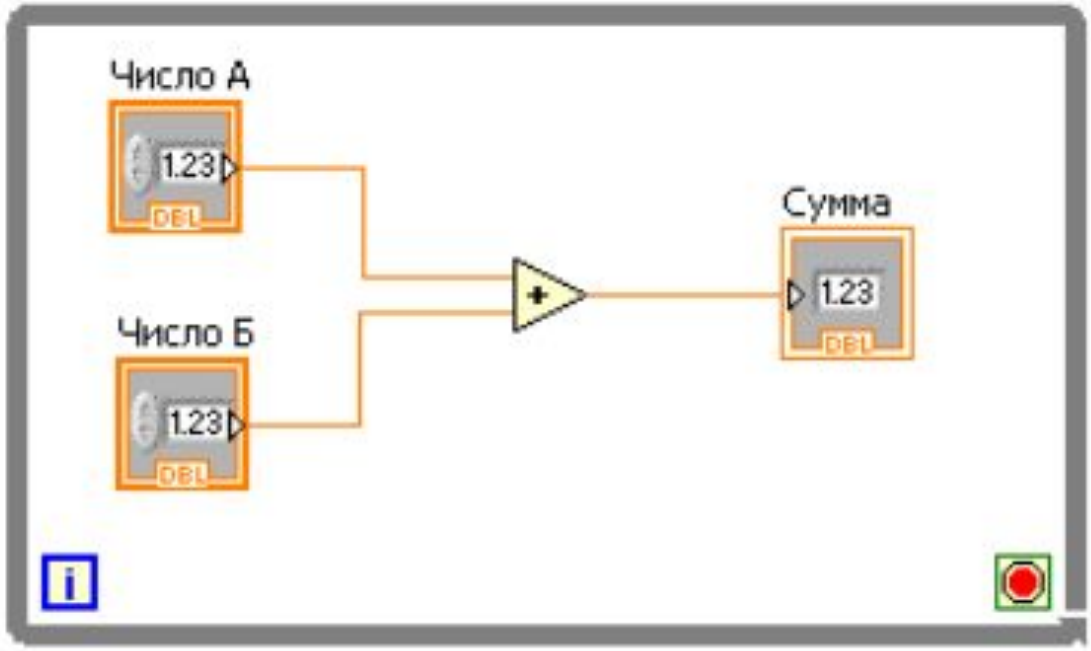
Структуры

- С помощью структур можно осуществить повторение отдельных частей программы, выполнение той или иной части программы в зависимости от какого-либо условия, выполнение программы в строго определенном порядке
- Functions => Structures



While Loop

Functions - Programming - Structures - While Loop



Loop Iteration (Повторение цикла). Имеет один контакт, расположенный ***справа*** – выход, значение которого соответствует количеству выполненных повторений цикла. ***Синий цвет*** терминала и контакта показывает, что выходное значение представлено целым числом.

а) 

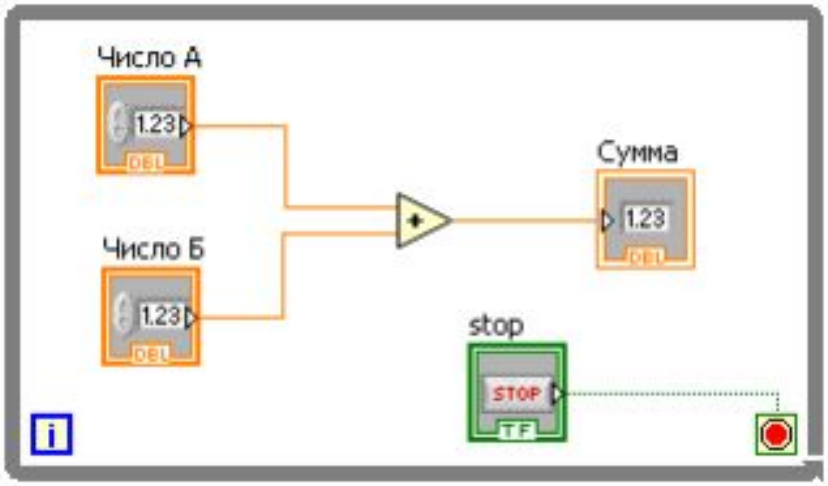
Loop Condition (Условие цикла) терминал условия выхода из цикла. Имеет один контакт ***слева*** – вход. Он может работать в двух режимах:

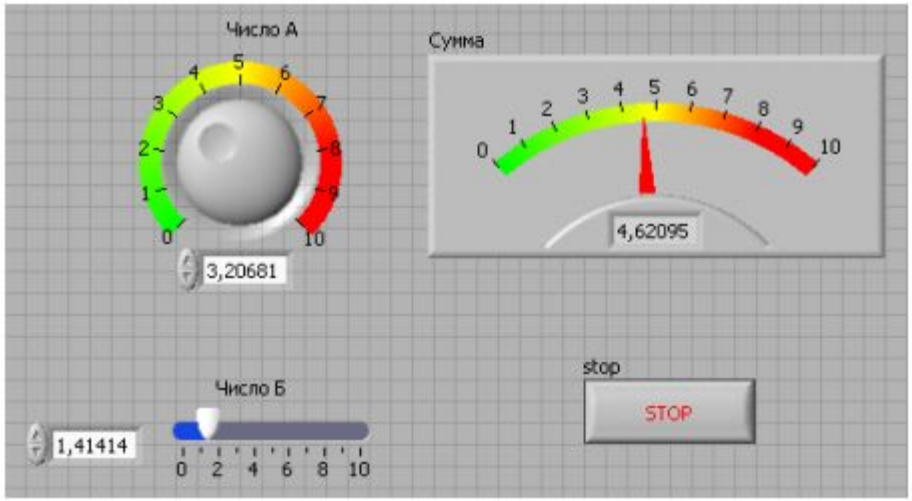
или

б) 

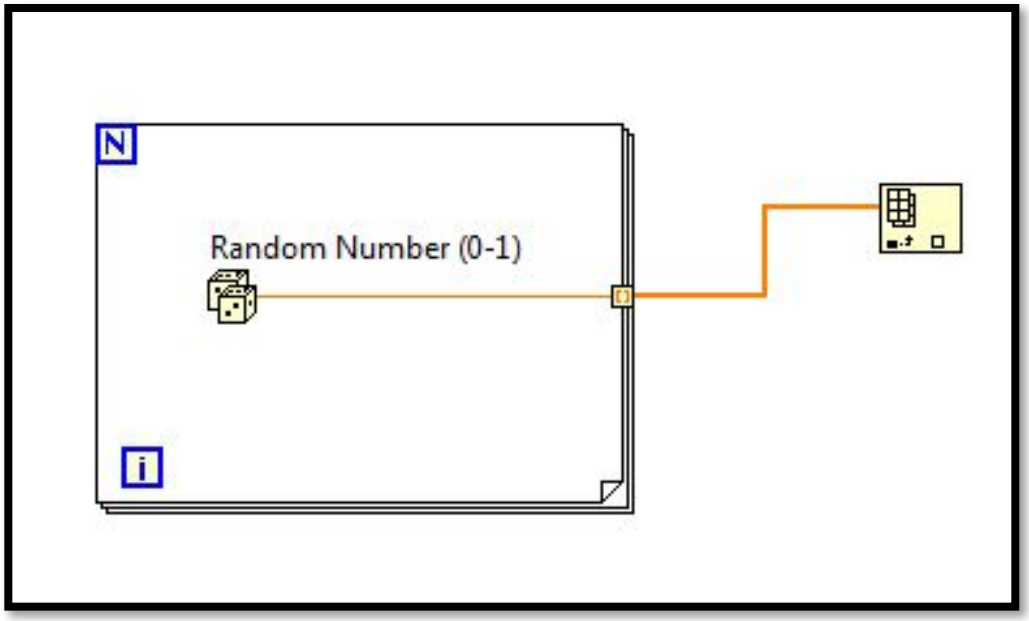
а) ***Stop if True*** (остановить, если Истинный) – цикл прерывается, если на этот терминал поступает значение булевой переменной ***True*** (Истинный).

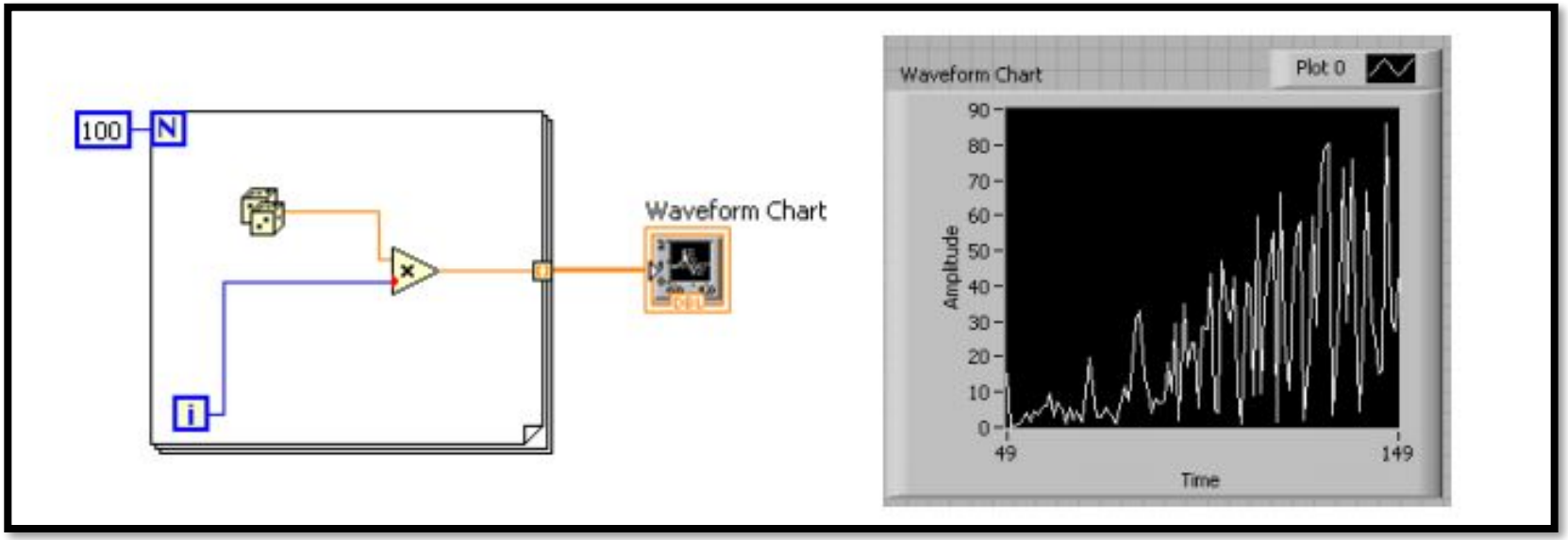
б) ***Continue if True*** (продолжать, если Истинный) – цикл продолжается до тех пор, пока значение этого терминала – ***True.*** Зеленый цвет терминала и контакта показывает, что формат данных, передаваемых ему, ***булев*** (логический).

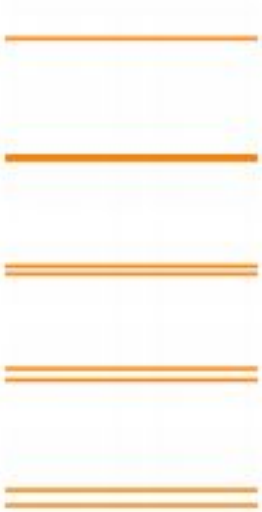

Число А
1.23
DBL
Сумма
1.23
DBL
Число Б
1.23
DBL
stop
STOP
TF


Число А
0
1
2
3
4
5
6
7
8
9
10
3,20681
Сумма
0
1
2
3
4
5
6
7
8
9
10
4,62095
Число Б
1,41414
0
2
4
6
8
10
stop
STOP

Генерация массива псевдослучайных чисел For Loop

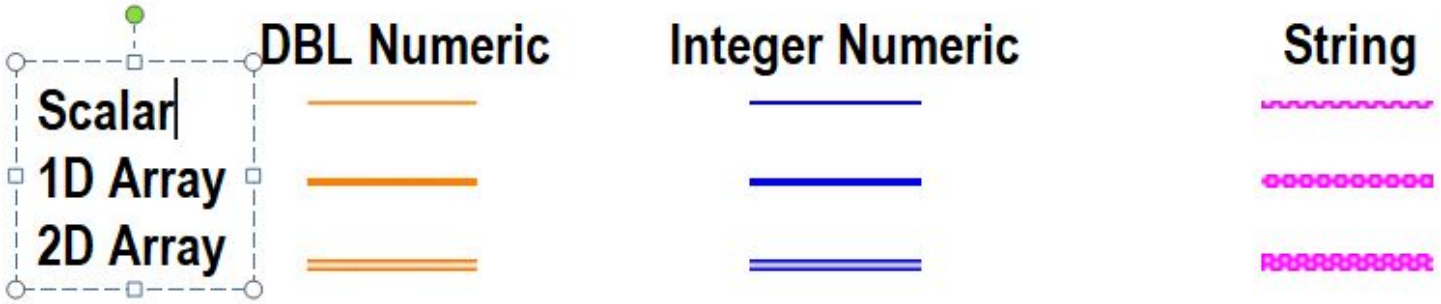



100
N
i
Waveform Chart
Waveform Chart
Plot 0
Amplitude
90
80
70
60
50
40
30
20
10
0
49
149
Time


Скаляр (одно число)
Одномерный массив (вектор)
Двумерный массив
Трехмерный массив
Четырехмерный массив

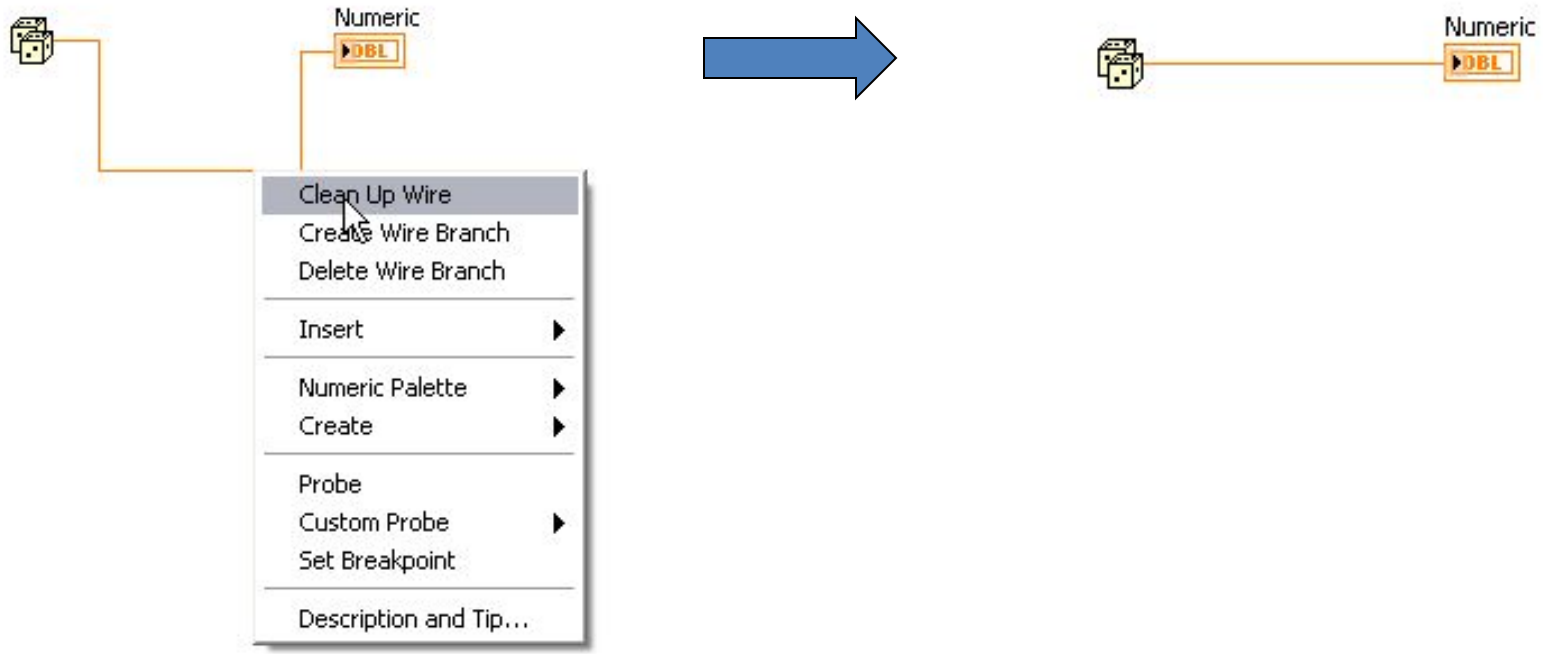
F. Block Diagram – Проводники

- Данные между объектами блок-диаграммы передаются по проводникам
- Проводники имеют разный стиль, цвет и толщину, которые зависят от типа данных
- Оборванный проводник выглядит, как черная пунктирная линия с красным X посередине



F. Block Diagram – Советы для соединений

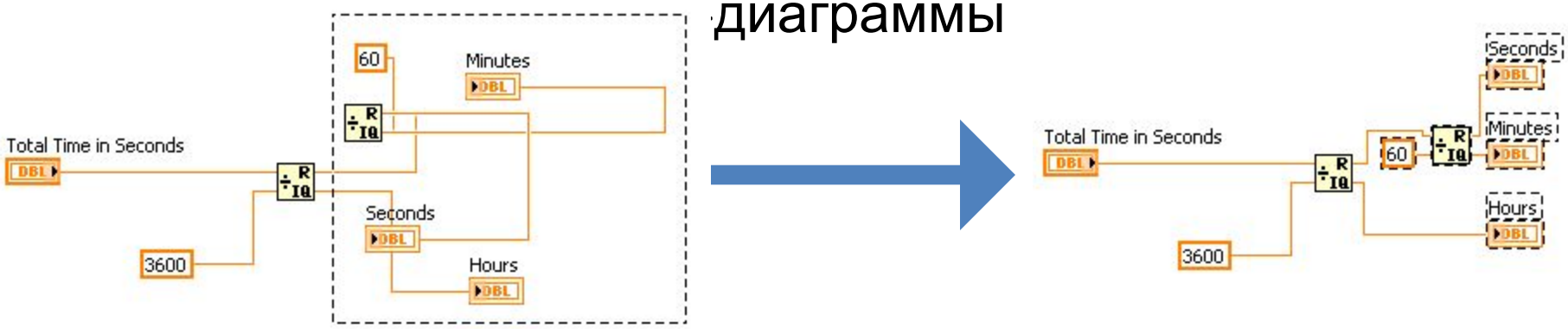
- Нажмите <Ctrl>-B, чтобы удалить все разорванные проводники
- Щелкните правой кнопкой мыши и выберите **Clean Up Wire** для изменения маршрута, по которому проходит проводник



F. Block Diagram – Советы для соединений

Используйте инструмент Clean Up Diagram (привести в порядок диаграмму) для упорядочения проводников и объектов с целью улучшения читаемости

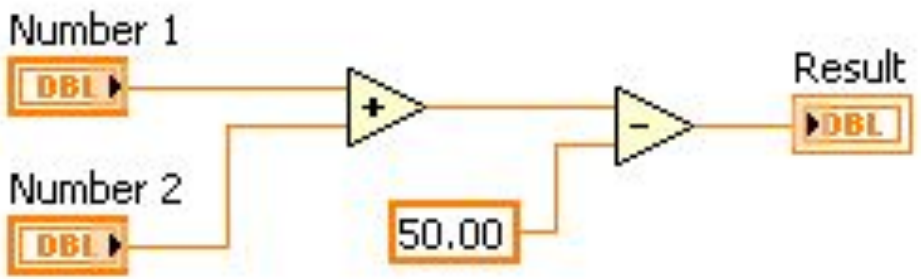
1. Выделите фрагмент блок-диаграммы
2. Щелкните по кнопке Clean Up Diagram на панели диаграммы



I. Потоковое программирование

LabVIEW использует модель потока данных для управления исполнением VI

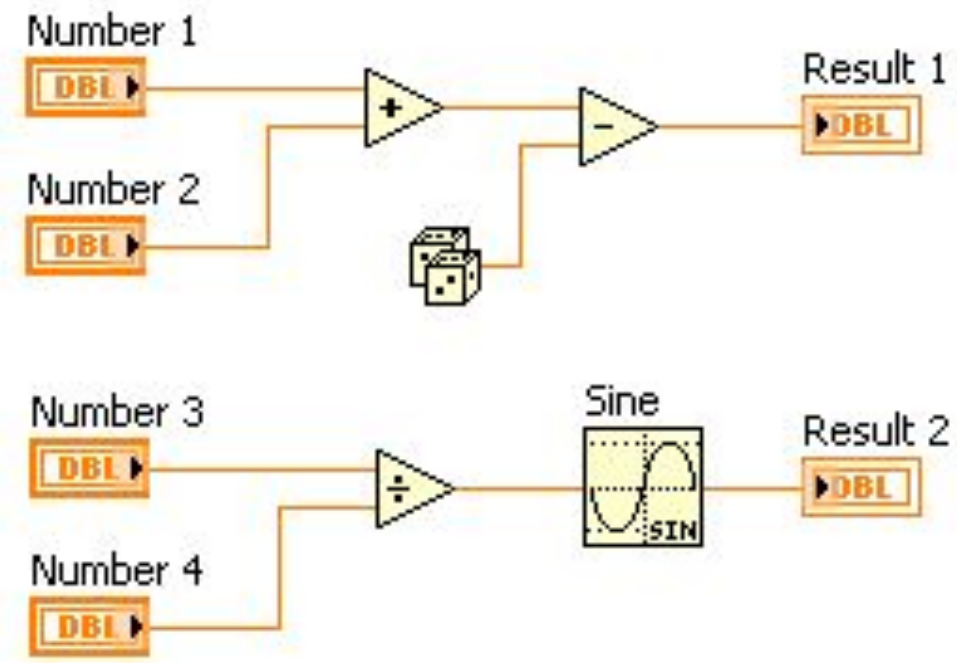
– Узел выполняется только, когда данные доступны на всех его входных терминалах
– Узел передает данные на выходные терминалы только когда завершается испол



I. Потоковое программирование – Контрольный вопрос

Какой узел выполняется первым?

a) Add
b) Subtract
c) Random Number
d) Divide
e) Sine



I. Потоковое программирование – Ответ на контрольный вопрос

НЕТ КОРРЕКТНОГО ОТВЕТА

Какой узел выполняется пер

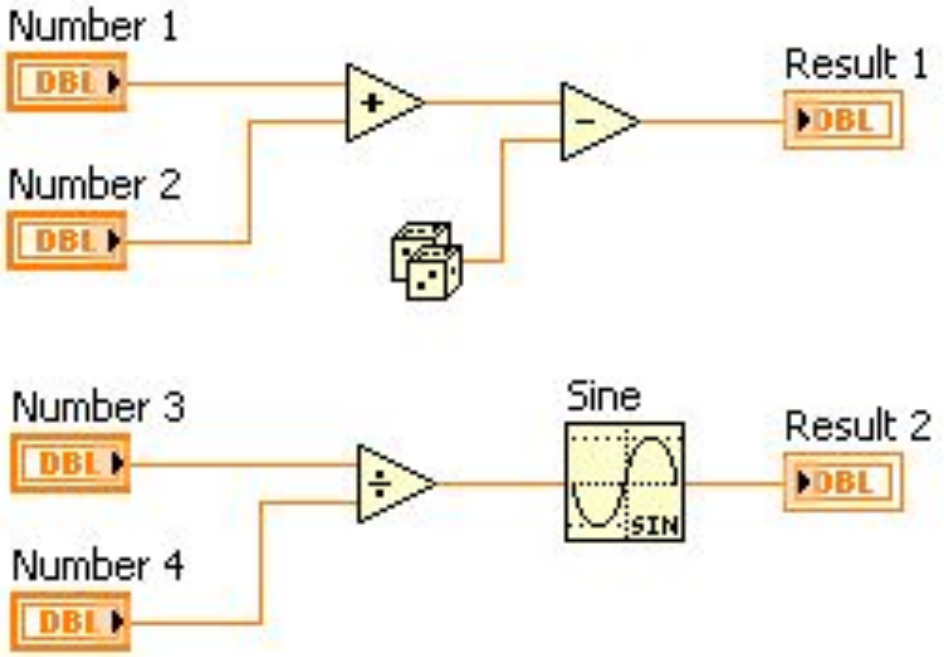
a) *Add – возможно*

b) *Subtract – определенно н*

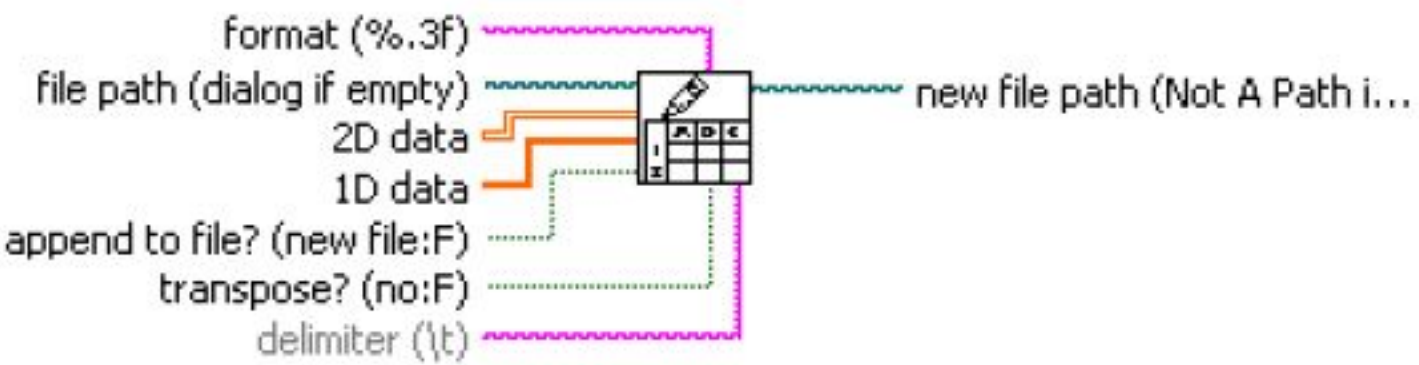
c) *Random Number – возмож*

d) *Divide – возможно*

e) *Sine – определенно нет*



Запись в файл
Functions - Programming - File I/O Write to Spreadsheet File);



Процедура ***Write to Spreadsheet File*** позволяет быстро сохранить интересующие нас данные в текстовый файл. При этом она автоматически выполняет несколько последовательных операций:
1) преобразует массив чисел в строку символов (ASCII кодов символов);
2) создает файл для записи этих чисел на жестком диске компьютера;
3) записывает строку символов в этот файл;
4) завершает работу с диском – закрывает файл.

Входы:

1. ***format (%.3f)*** – формат (строка, определяющая форматирование выводимых данных). Тип данных – ***String*** (строковая переменная) – сиреневый проводник. По умолчанию принимает значение: **%** – ключ строки формата, **.3** – три знака после запятой, ***f*** – формат с плавающей точкой.

2. ***file path (dialog if empty)*** – путь к файлу (если не подключен – показывает диалоговое окно «Сохранить файл»). Тип данных – ***Path*** (путь к файлу, ссылка) – сине-зеленый провод.

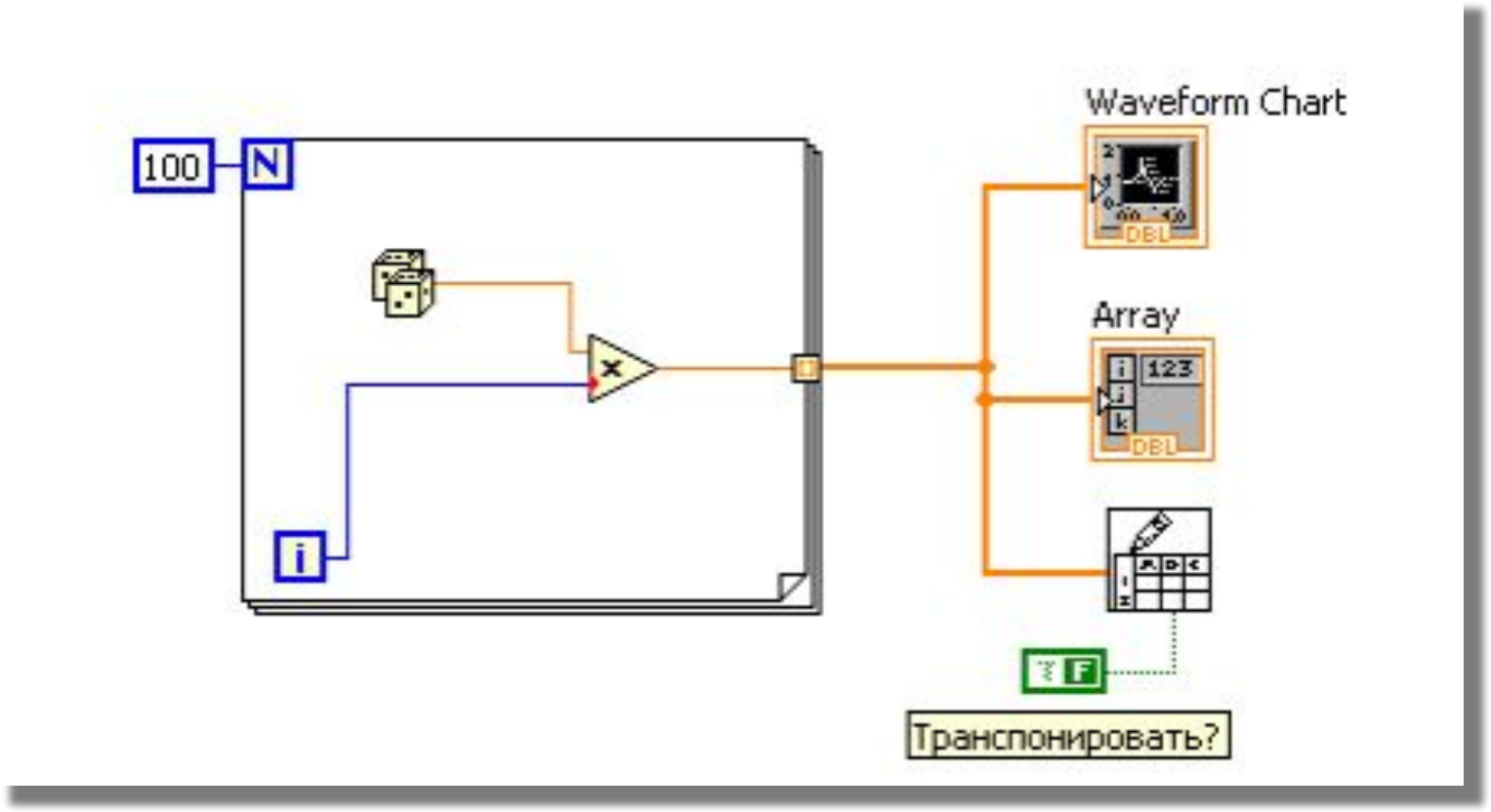
3. ***1D Data*** – одномерные данные (одномерный массив, вектор) 1D от английского 1 Dimension – одно измерение. Тип данных – ***Double*** – оранжевый провод.

4. ***2D Data*** – двумерные данные. Тип данных – ***Double*** – оранжевый провод.

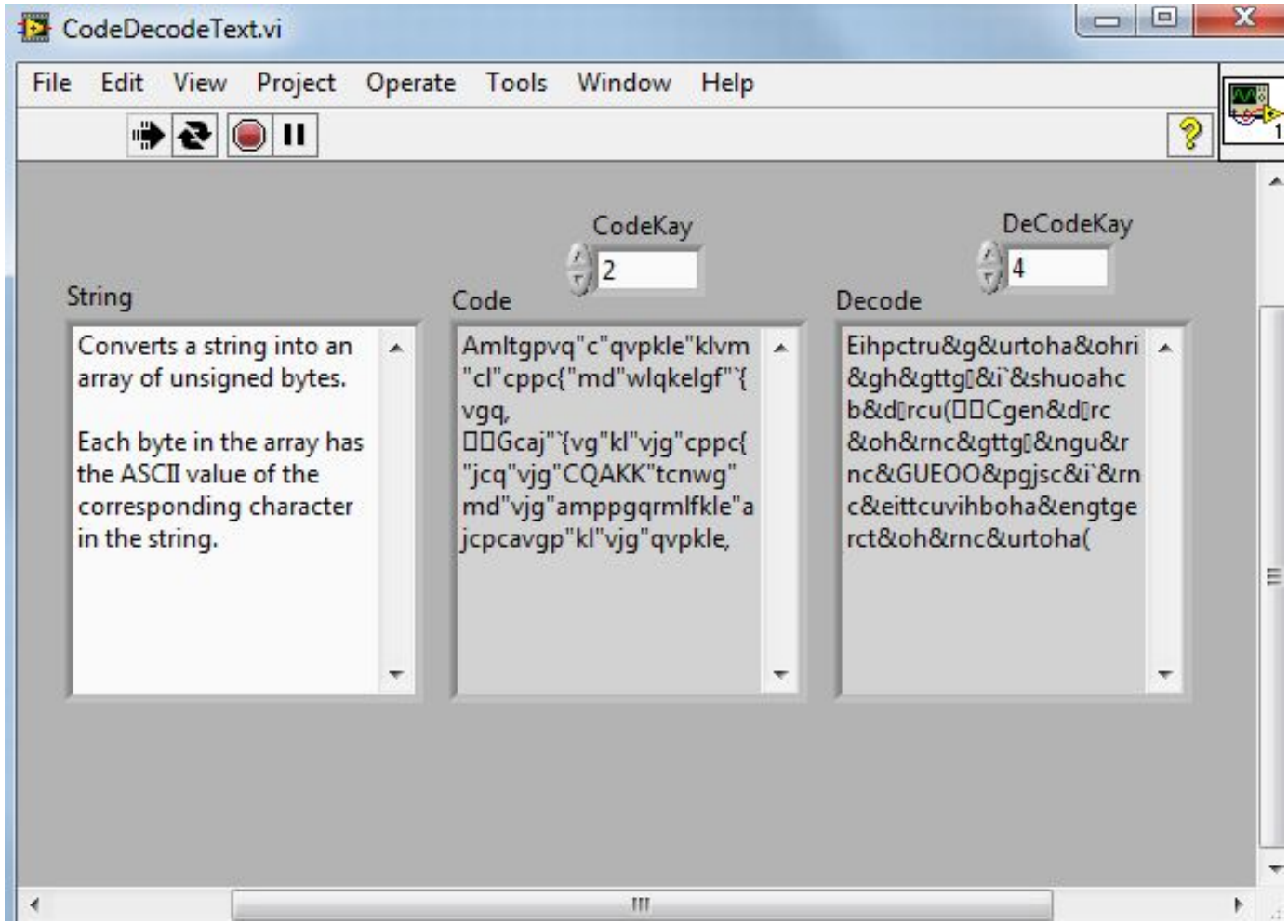
5. ***append to file? (new file:F)*** – добавить к (существующему) файлу? (по умолчанию – создать новый файл, ***False***). Тип данных – ***Boolean*** – зеленый провод и знак вопроса в названии терминала.

6. ***transpose?(no:F)*** – транспонировать? (по умолчанию – нет, ***False***). Тип данных – ***Boolean*** – зеленый провод и знак вопроса в названии терминала.

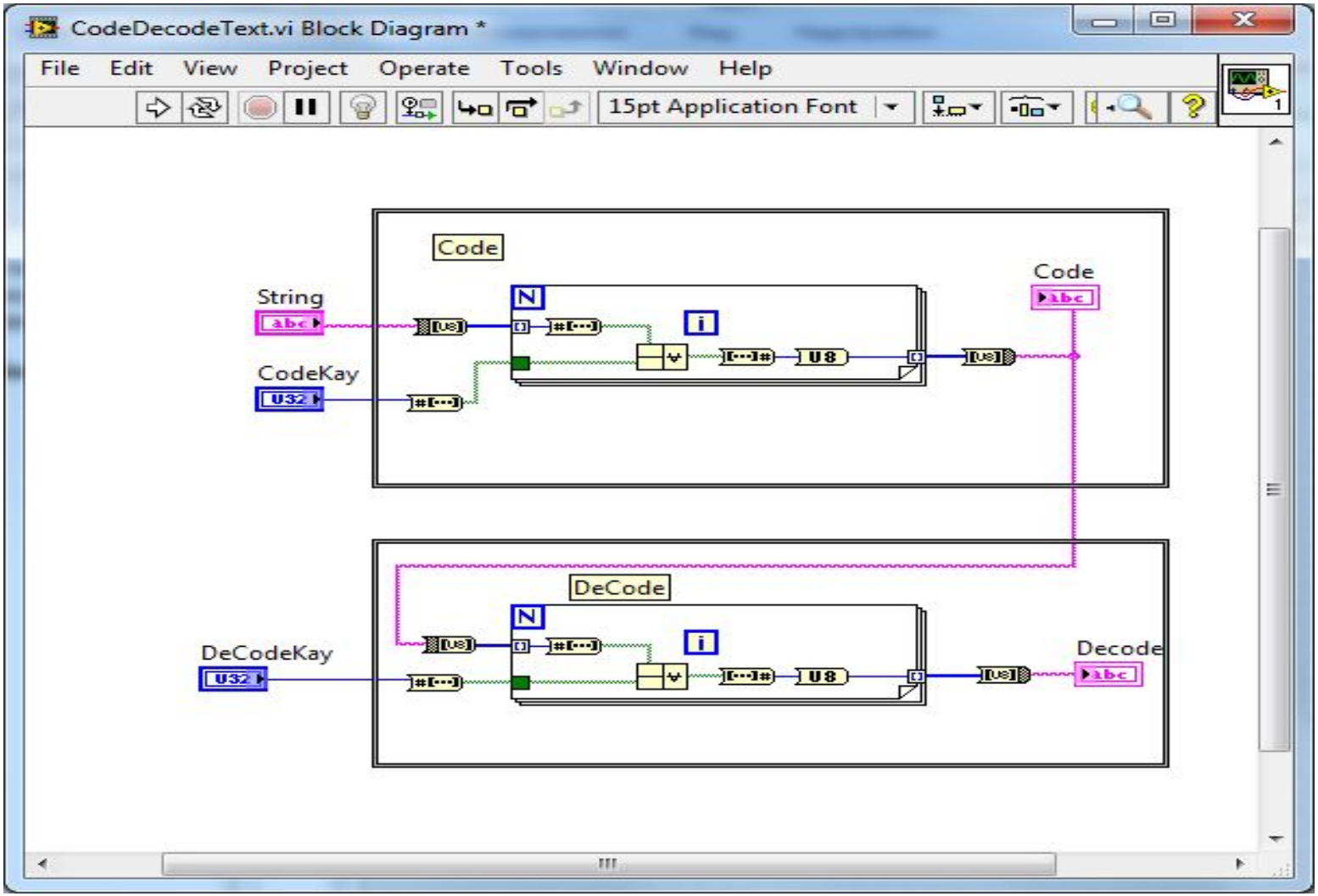
7. ***delimiter(\t)*** – разделитель (столбцов в таблице) (по умолчанию символ табуляции ***Tab,*** показан в принятом в LabVIEW backslash code (коде обратной косой черты) – \t). Тип данных – ***String*** (строковая переменная) – сиреневый проводник.


100
N
i
Waveform Chart
DBL
Array
123
DBL
Транспонировать?

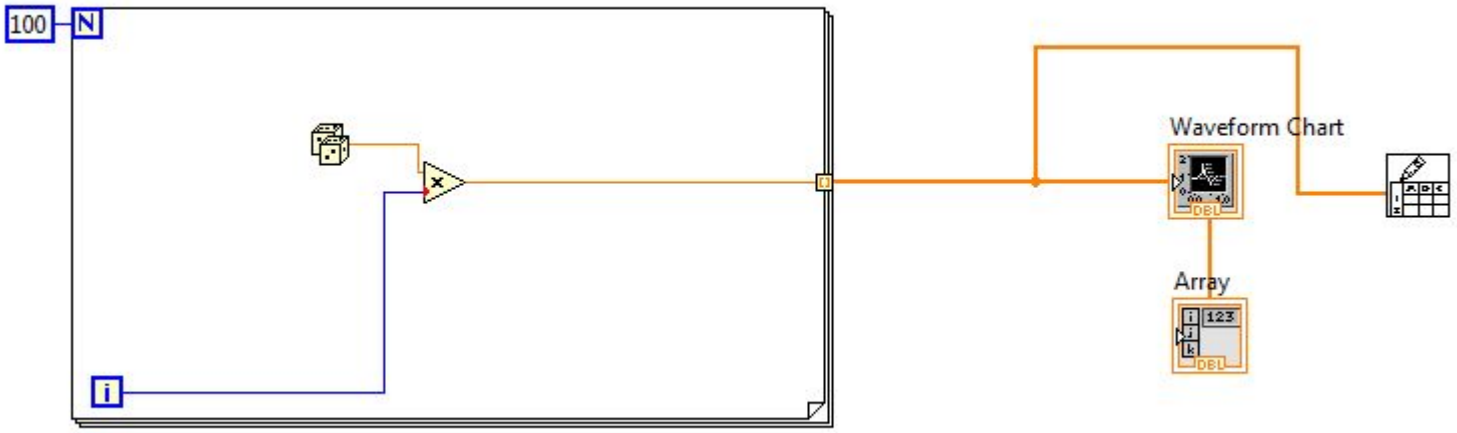
Пример кодирования



Блок диаграмма

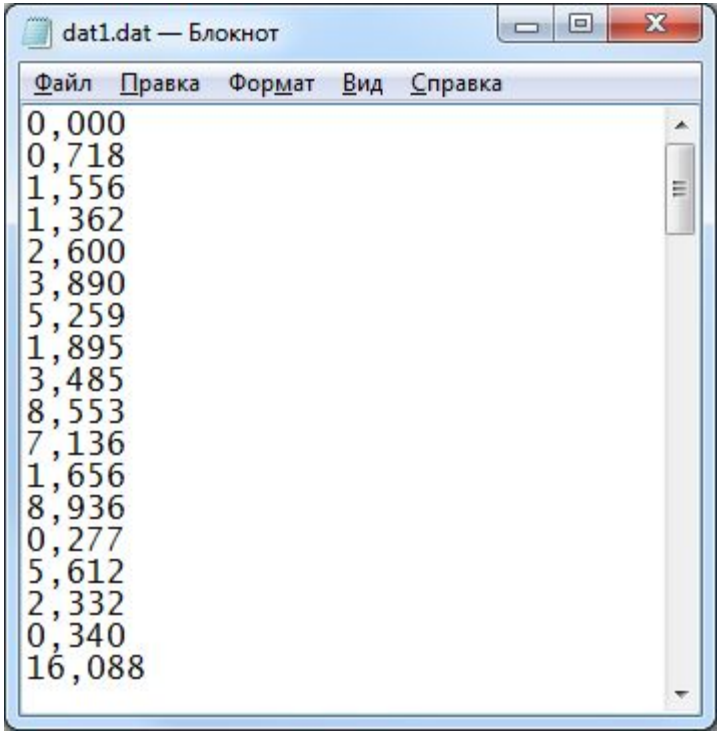


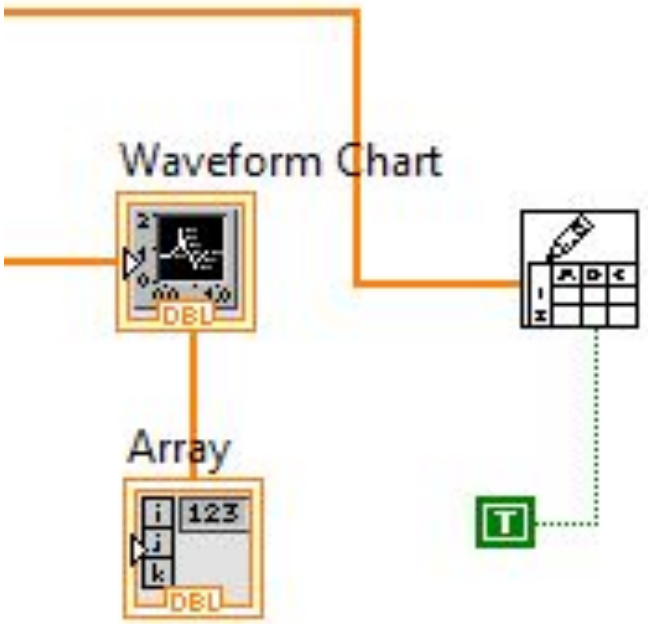
Подключение данных



Константа для транспонирования

- ***(Programming >Boolean > False constant)***





Элементы Языка C/C++

Файлы

- Текстовый файл – файл, содержащий текст, разбитый на строки парой специальных кодов: «возврат каретки» (0x13) и «перевод строки» (0x10). Если файл открыт в текстовом режиме, то при чтении из такого файла комбинация этих кодов преобразуется в один символ’\n’ — переход к новой строке. При записи в файл осуществляется обратное преобразование.
- Бинарный файл – файл, из которого байты считываются и выводятся в первоначальном виде без каких-либо преобразований. Если требуется указать на такой файл, то к параметру добавляется буква b. Например: rb, или wb, или r+b. В некоторых компиляторах текстовый режим обмена обозначается буквой t, т.е. записывается r+t или rt.

Функции чтения из файла и записи в файл:

fputc(переменная типа char, указатель на файл) – посимвольная запись данных в файл

fgetc(указатель на файл) – посимвольное чтение из файла

fputs(переменная типа строка, указатель на файл) – построчная запись данных в файл. Записывает в файл строку, но в конце не добавляет символ окончания строки.

fgets(переменная типа строка, длина, указатель на файл) – построчное чтение данных из файла. Читает строку целиком до символа новой строки, если ее длина не превышает значения параметра «длина» минус один символ. Параметр «длина» является целым числом или целочисленной переменной,указывающей максимально возможное количество символов в строке

fprintf(указатель на файл, строка формата, список переменных) – форматированный вывод символов, строк или чисел в файл

fwrite(указатель на буфер хранения данных, размер элемента, количество элементов, указатель на файл) – запись заданного количества блоков данных определѐнной длины из буфера в файл

fscanf(указатель на файл, строка формата, список переменных) – форматированный ввод символов строк или чисел из файла.

fread(указатель на буфер размещения данных, размер элемента, количество элементов, указатель на файл) – чтение блоков данных заданного размера в указанном количестве из файла в буфер.

feof(указатель на файл) – функция определяет, достигнут ли конец файла. Если текущая позиция является концом файла (EOF), то функция возвращает ненулевое значение, в противном случае возвращается 0.

fflush(указатель на файл) – принудительная очистка буфера вывода путем передачи содержимого на ВЗУ

remove(имя файла) – удаляет файл. Функция remove() возвращает 0, если файл успешно удален

rename(старое имя, новое имя) – переименовывает файл или директорию, указанную в параметре «старое имя», и присваивает имя, указанное в параметре «новое имя». Также может применяться для перемещения файла.

fseek(указатель на файл, количество байт, начало отсчѐта) -- устанавливает указатель текущей позиции в файле. Количество байт отсчитывается от значения параметра «начало отсчета», оно определяет новое значение указателя текущей позиции, а начало отсчѐта - это один из следующих макросов: начало файла (SEEK_SET), текущая позиция (SEEK_CUR), конец
файла (SEEK_END). Обычно данная функция применяется только для бинарых файлов.

Пример

- В программе создать текстовый файл ft, содержащий n случайных целых чисел. Считать числа из ft, подсчитать среднее значение всех чисел и записать в бинарный файл fd все числа меньше этого среднего значения.

```c
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <conio.h>
#define A 1
#define B 100
int main(void)
{
    FILE *in,*out;
    int i,s,n,c,*a;
    printf("N=");
    scanf("%d",&n);
    if((in=fopen("ft.dat","w"))==NULL)
    {
        printf("Cannot open input file.\n");
        return 1;
    }
    srand(time(NULL));
    for(i=0;i<n;i++)
    {
        if(i==(n-1))
            fprintf(in,"%d",A+rand()%B);
        else
            fprintf(in,"%d\n",A+rand()%B);
    }
    fclose(in);
```

```
a=(int*)calloc(sizeof(int),n);
if((in=fopen("ft.dat","r"))==NULL)
{
    printf("Cannot open input file.\n");
    return 1;
}
i=0;s=0;
if(a!=NULL)
{
    while(!feof(in))
    {
        fscanf(in,"%d",&a[i]);
        s+=a[i];
        i++;
    }
}
else
{
    printf("OPS\n");
    return 2;
}
s=(int)(s*1.0/n+0.5);
printf("%d\n",s);
if((out=fopen("fb.dat","wb"))==NULL)
{
    printf("Cannot open input file.\n");
    return 1;
}
for(i=0;i<n;i++)
{
    if(s>a[i])
    {
        fwrite(a+i,sizeof(int),1,out);
        printf("%d\n",a[i]);
    }
}
fclose(in);free(a);fclose(out);getch();
return 0;
```

```
#include <time.h>
#include <iostream>
#include <fstream>
#define A 1
#define B 100
using namespace std;
int main(void)
{
    int i,s,n;
    setlocale(LC_ALL, "rus");
    cout<<"Введите количество чисел:";
    cin>>n;
    ofstream fout("ft.dat");
    if(!fout)
    {
        cout<<"\n Нельзя создать файл";
        return 1;
    }
    srand(time(NULL));
    for(i=0;i<n;i++)
    {
        if(i==(n-1))
            fout<<(A+rand()%B);
        else
            fout<<(A+rand()%B)<<"\n";
    }
    fout.close();
```

```
    int *mas = new int [n];
    ifstream fin("ft.dat");
    s=0;
    for (i=0;i<n;i++)
    {
        fin>>mas[i];
        s+=mas[i];
    }
    fin.close();
    s=(int)(s*1.0/n+0.5);
    printf("S=%d\n",s);
    ofstream fout1("fb.dat",ios::binary|ios::out);
     if(!fout1)
    {
        cout<<"\n Нельзя создать файл";
        return 1;
    }
    for(i=0;i<n;i++)
    {
        if(s>mas[i])
        {
            fout1.write((char *) &mas[i], sizeof(int));
            cout<<mas[i]<<"\n";
        }
    }
    fout1.close();free(mas);system("pause");
    return 0;
}
```

```
/*Чтение данных из файла в массив. Количество данных
неизвестно.  Файл открывается дважды:
сначала для вычисления количество элементов в файле,
а затем - для считывания данных в динамический массив.*/

char * fileName="ft.dat";
ifstream fin(fileName);
int r,n=0;
while (!fin.eof())
{
fin>>r; n++;
}
fin.close();
int *mx= new int [n];
ifstream fin2(fileName);
for (int i=0;i<n;i++) fin2>>mx[i];
fin2.close();
for (int i=0;i<n;i++) cout<<mx[i]<<" ";
```

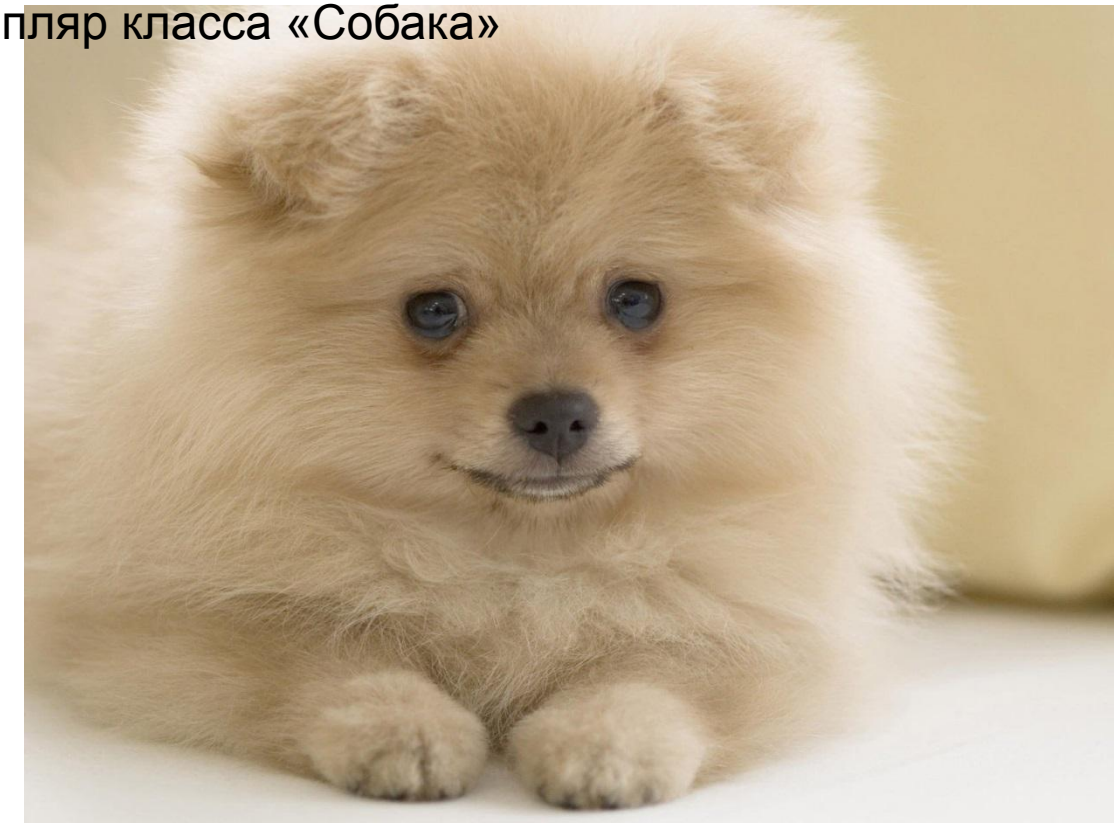
Основы ООП

```
private: void GoHarvest() {
        int xx,yy,rr,aa;
        if(X==TargetX&&Y==TargetY) {
                rr=DrainOre(X,Y,(MaxLoad-Load));
                Load+=rr;
                TotalLoad+=rr;
                GatheredSomething+=rr;
                for(int rr=1; rr<15; rr++) {
                        for(float aa=0; aa<=2.1; aa+=(.19/rr)) {
                                xx = (X + (rr*sin(aa*Pi))+.5);
                                yy = (Y + (rr*cos(aa*Pi))+.5);
                                if((xx>=0&&xx<320)&&(yy>=0)&&(yy<200)&&pget(xx,yy)<250) pset(xx,yy,OreValue(xx,yy));
                        }
                }
        } else {
                pset(TargetX,TargetY,254);
                MoveTo(TargetX,TargetY);
                if(TargetX<X) {
                        xx=TargetX;
                } else {
                        xx=(320-TargetX);
                }
                if(TargetY<Y) {
                        yy=TargetY;
                } else {
                        yy=(200-TargetY);
                }
                HarvDist=sqrt(pow(xx,2)+pow(yy,2));
        }
}
public: int Status() {
        return status;
}
public: int LLoad() {
        return LastLoad;
}
public: int TLoad() {
        return TotalLoad;
```

Парадигма программирования, основанная на представлении предметной области в виде взаимосвязанных **абстрактных объектов** и их **реализаций**

Классы и объекты

- В ООП вводится понятие **Класса** – пользовательского типа данных, объединяющего **данные** и **методы** их обработки
- **Объектом** называется **экземпляр** класса
  - Собака – это класс
  - Собака Жучка из 3 подъезда – это объект, представитель или экземпляр класса «Собака»



```
driveTo(you, work);
```

```
you.driveTo(work);
```

Объекты имеют два основных компонента:

Список соответствующих свойств (например: вес, цвет, размер, прочность, форма и т.д.).

Поведение, которое они могут проявлять (например: открывать что-либо, делать что-то и т.д.).

```cpp
#include <iostream>

struct DateStruct
{
    int day;
    int month;
    int year;
};

void print(DateStruct &date)
{
    std::cout << date.day<< "/" << date.month << "/" << date.year;
}

int main()
{
    DateStruct today { 12, 11, 2018}; // используем uniform инициализацию

    today.day = 18; // используем оператор выбора члена для выбора члена структуры
    print(today);

    return 0;
}
```

```cpp
#include <iostream>

class DateClass
{
public:
    int m_day;
    int m_month;
    int m_year;

    void print()
    {
        std::cout << m_day << "/" << m_month << "/" << m_year;
    }
};

int main()
{
    DateClass today { 12, 11, 2018 };

    today.m_day = 18; // используем оператор выбора членов для выбора переменной-члена объекта today класса DateClass
    today.print(); // используем оператор выбора членов для вызова метода объекта today класса DateClass

    return 0;
}
```

```cpp
#include <iostream>
#include <string>

class Employee
{
public:
    std::string m_name;
    int m_id;
    double m_wage;

    // Метод вывода информации о работнике на экран
    void print()
    {
        std::cout << "Name: " << m_name <<
                "\nId: " << m_id <<
                "\nWage: $" << m_wage << '\n';
    }
};

int main()
{
    // Объявляем два работника
    Employee john { "John", 5, 30.00 };
    Employee max { "Max", 6, 32.75 };

    // Выводим информацию о работнике на экран
    john.print();
    std::cout<<std::endl;
    max.print();

    return 0;
}
```

Данные объекта (переменные объекта, члены-данные)

- **Члены-данные** (data members) хранят всю необходимую информацию об объекте, формируют его состояние, характеристики и т.п.
  - Изменение состояния объекта или его характеристик связано с изменением данных, в нем содержащихся

Методы класса

- Класс может содержать один или более **методов**, позволяющих осуществлять манипуляцию данными объекта
- **Метод объекта** – программный код, выполненный в виде процедуры или функции, реагирующий на передачу объекту определенного сообщения
- Вызов метода объекта может приводить к изменению его состояния (значение членов-данных), а может и не приводить
  - Пример 1: поиск и замена текста в документе
  - Пример 2: проверка правописания текста документа

Пример: Треугольник

- Свойства
  - Координаты вершины A
  - Координаты вершины B
  - Координаты вершины C
  - Площадь
  - Периметр
  - Координаты центра вписанной окружности
- Методы
  - Переместить в заданном направлении
  - Отмасштабировать
  - Повернуть вокруг заданной точки

main.c F9

```
1  class Point
2  {
3  public:
4      double x, y;
5  };
6
7  class Triangle
8  {
9  public:
10     double GetArea();
11     double GetPerimeter();
12     Point GetCenter();
13
14     void Move(double dx, double dy);
15     void Scale(double sx, double sy);
16     void Rotate(Point center, double angle);
17     Point p0, p1, p2;
18 };
19
```

Важнейшие принципы ООП

- Абстракция данных
- Инкапсуляция
- Наследование
- Полиморфизм

Абстракция данных

- Объекты представляют неполную информацию о реальных сущностях предметной области
  - Абстракция позволяет оперировать с объектном на уровне, адекватном решаемой задаче
  - Высокоуровневые обращения к объекту могут обрабатываться с помощью вызова функций и методов низкого уровня

Инкапсуляция

- **Инкапсуляция** - способность объекта скрывать внутреннее устройство своих свойств и методов
  - Согласно данному принципу, класс должен рассматриваться как **черный ящик**
    - Внешний пользователь не знает детали реализации объекта и работает с ним только путем предоставленного объектом **интерфейса**
  - Следование данному принципу может уменьшить число связей между классами и упростить их независимую реализацию, модификацию и тестирование

Ограничение доступа к данным и методам класса

- Доступ к данным и методам класса извне может быть ограничен
  - Рекомендуется запрещать доступ к данным класса в обход его методов
- Для разделения прав доступа к полям класса используются ключевые слова
  - **public:**
  - **private:**
  - **protected:**

Публичные (public) поля класса

- Public-методы и данные класса определяют его интерфейс
  - доступ к ним возможен из любой части кода
  - необходимо помещать в public-раздел класса только необходимый набор методов, выполняющих высокоуровневые операции над объектом класса

Закрытые (частные) поля класса

- Private-данные и методы класса определяют его реализацию
  - Доступ к ним разрешен только из методов данного класса
  - **Рекомендуется все данные класса делать закрытыми,** их обработку осуществлять внутри методов
  - Закрытые методы класса обычно используются публичными методами, решая внутренние задачи класса

Защищенные поля класса

- Protected-данные и методы определяют интерфейс для производных классов
  - Доступ к ним разрешен изнутри методов данного класса и всех его потомков
  - В защищенной зоне размещают методы, которые не должны быть видны снаружи класса, но реализация которых может быть переопределена или использована производными классами

```
class DateClass // члены класса являются закрытыми по умолчанию
{
   int m_day; // закрыто по умолчанию, доступ имеют только другие члены класса
   int m_month; // закрыто по умолчанию, доступ имеют только другие члены класса
   int m_year; // закрыто по умолчанию, доступ имеют только другие члены класса
};

int main()
{
   DateClass date;
   date.m_day = 12; // ошибка
   date.m_month = 11; // ошибка
   date.m_year = 2018; // ошибка

   return 0;
}
```

```
class DateClass
{
public: // Спецификатор доступа
    int m_day; // открыто, доступ имеет любой объект
    int m_month; // открыто, доступ имеет любой объект
    int m_year; // открыто, доступ имеет любой объект
};

int main()
{
    DateClass date;
    date.m_day = 12; // ок, так как m_day имеет спецификатор доступа public
    date.m_month = 11;  // ок, так как m_month имеет спецификатор доступа
public
    date.m_year = 2018;  // ок, так как m_year имеет спецификатор доступа public

    return 0;
}
```

```cpp
#include <iostream>

class DateClass // члены класса являются закрытыми по умолчанию
{
    int m_day; // закрыто по умолчанию, доступ имеют только другие члены класса
    int m_month; // закрыто по умолчанию, доступ имеют только другие члены класса
    int m_year; // закрыто по умолчанию, доступ имеют только другие члены класса

public:
    void setDate(int day, int month, int year) // открыто, доступ имеет любой объект
    {
        // метод setDate() имеет доступ к закрытым членам класса, так как сам является членом класса
        m_day = day;
        m_month = month;
        m_year = year;
    }

    void print() // открыто, доступ имеет любой объект
    {
        std::cout << m_day << "/" << m_month << "/" << m_year;
    }
};

int main()
{
    DateClass date;
    date.setDate(12, 11, 2018); // ок, так как setDate() имеет спецификатор доступа public
    date.print(); // ок, так как print() имеет спецификатор доступа public

    return 0;
}
```

Наследование

- **Наследование** позволяет описать новый класс на основе уже существующего **родительского** (базового) класса
  - Класс-потомок может добавить свои собственные свойства и методы, пользоваться методами и свойствами базового класса
  - Наследование позволяет строить иерархии классов

Пример

```
class Plane
{
public:
    void TakeOff();
    void Fly();
    void Land();
private:
    double m_fuel;
};

class MilitaryPlane : public Plane
{
public:
    void Attack();
private:
    int m_ammo;
};
```

Полиморфизм

- **Полиморфизмом** называют явление, при котором классы-потомки могут изменять реализацию метода класса-предка, сохраняя его интерфейс
  - Полиморфизм позволяет обрабатывать объекты классов-потомков как однотипные объекты, не смотря на то, что реализация методов у них может различаться

```
class Shape
{
public:
    virtual double GetArea()=0;
};

class Rectangle : public Shape
{
public:
    virtual double GetArea()
    {
        return width * height;
    }
private:
    double  width, height;
};

class Circle : public Shape
{
public:
    virtual double GetArea()
    {
        return 3.1415927 * radius * radius;
    }
private:
    double radius;
};
```

Размещение классов в различных файлах

- Общепринятой практикой является размещение объявления классов в заголовочных файлах .h, а их реализации – в файлах .cpp
  - Повышение модульности проекта
  - Каждый класс может быть подключен для дальнейшего использования при помощи директивы #include “имя заголовочного файла”
  - При внесении изменений в реализацию метода класса перекомпиляции подвергнутся только измененные файлы

Пример

date.h

```
class Date
{
public:
     void Next();
     void Print();
private:
     int m_day;
     int m_month;
     int m_year;
};
```

date.cpp

```
#include “date.h”

void Date::Next()
{
     // ...
}

void
Date::Print()
{
     // ...
}
```

main.cpp

```
#include “date.h”

int main()
{
     Date date1;
     return 0;
}
```

Инициализация экземпляра класса

- Для инициализации состояния объекта в момент его создания существует специальная функция – **конструктор**
  - Конструктор имеет то же имя, что и имя класса
  - Тип возвращаемого значения для конструктора не указывается (даже void)
  - Конструктор вызывается в момент создания экземпляра класса (объявление переменной класса или вызов оператора new)
  - Класс может иметь несколько конструкторов, предоставляющих различные способы инициализации объекта

```cpp
class Boo
{
public:
    int m_a;
    int m_b;
};

int main()
{
    Boo boo1 = { 7, 8 }; // список инициализаторов
    Boo boo2 { 9, 10 }; // uniform инициализация (C++11)

    return 0;
}
```

Конструктор по умолчанию

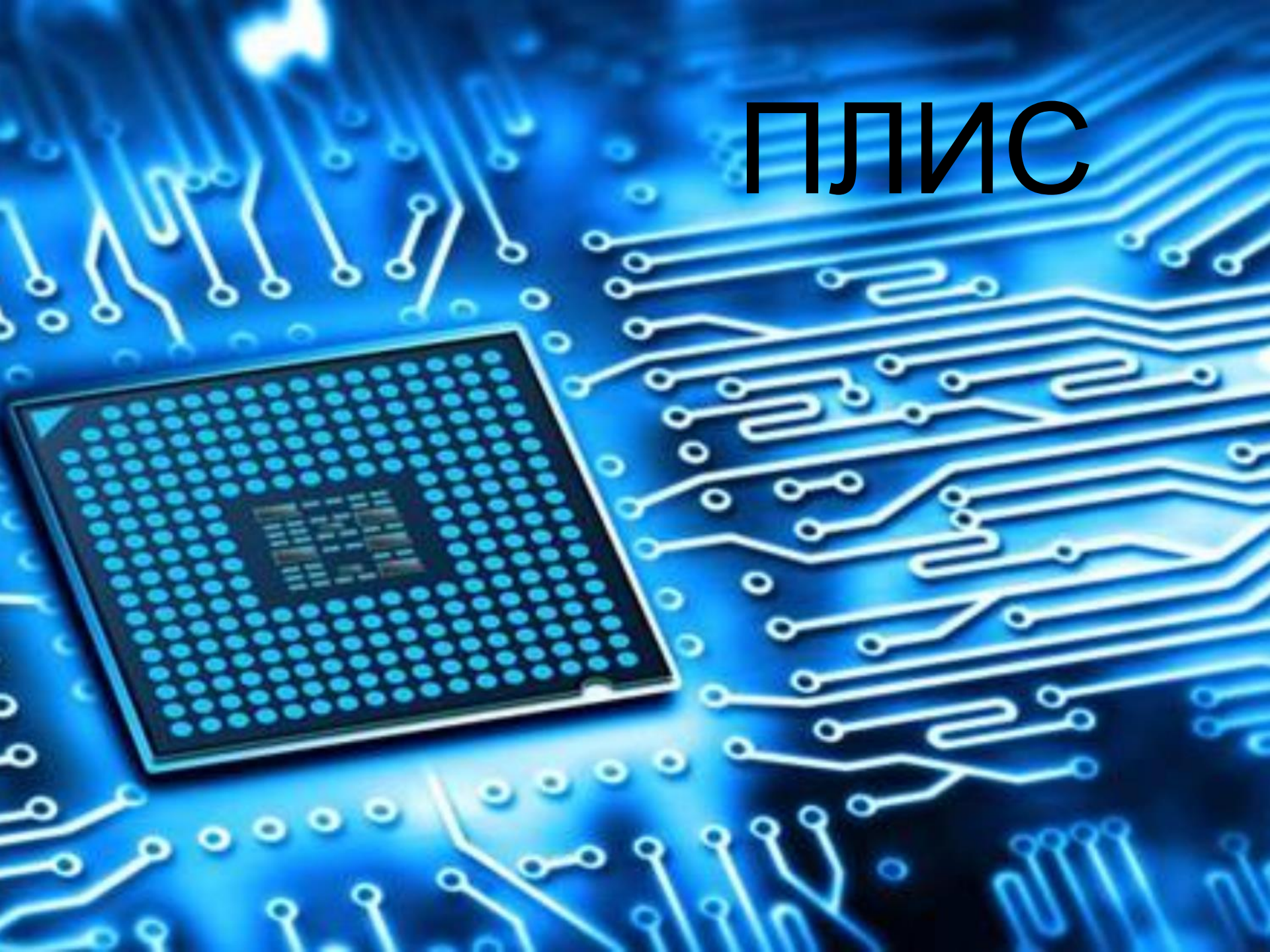
- Конструктор, не имеющий параметров, называется конструктором по умолчанию
  - Поля данных в таком конструкторе инициализируются значениями по умолчанию
  - Создавать такой конструктор или не создавать – зависит от конкретной задачи

```cpp
#include <iostream>

class Fraction
{
private:
    int m_numerator;
    int m_denominator;

public:
    Fraction() // конструктор по умолчанию
    {
        m_numerator = 0;
        m_denominator = 1;
    }

    int getNumerator() { return m_numerator; }
    int getDenominator() { return m_denominator; }
    double getValue() { return static_cast<double>(m_numerator) / m_denominator; }
};

int main()
{
    Fraction drob; // так как нет никаких аргументов, то вызывается конструктор по умолчанию Fraction()
    std::cout << drob.getNumerator() << "/" << drob.getDenominator() << '\n';

    return 0;
}
```

```cpp
#include <cassert>

class Fraction
{
private:
    int m_numerator;
    int m_denominator;

public:
    Fraction() // конструктор по умолчанию
    {
        m_numerator = 0;
        m_denominator = 1;
    }

    // Конструктор с двумя параметрами, один параметр имеет значение по умолчанию
    Fraction(int numerator, int denominator=1)
    {
        assert(denominator != 0);
        m_numerator = numerator;
        m_denominator = denominator;
    }

    int getNumerator() { return m_numerator; }
    int getDenominator() { return m_denominator; }
    double getValue() { return static_cast<double>(m_numerator) / m_denominator; }
};
```

```
int a(7); // инициализируем напрямую
Fraction drob(4, 5); // инициализируем напрямую, вызывается конструктор Fraction(int, int)


int a(7); // инициализируем напрямую
Fraction drob(4, 5); // инициализируем напрямую, вызывается конструктор Fraction(int, int)

int a { 7 }; // uniform инициализация
Fraction drob {4, 5}; // uniform инициализация, вызывается конструктор Fraction(int, int)
```

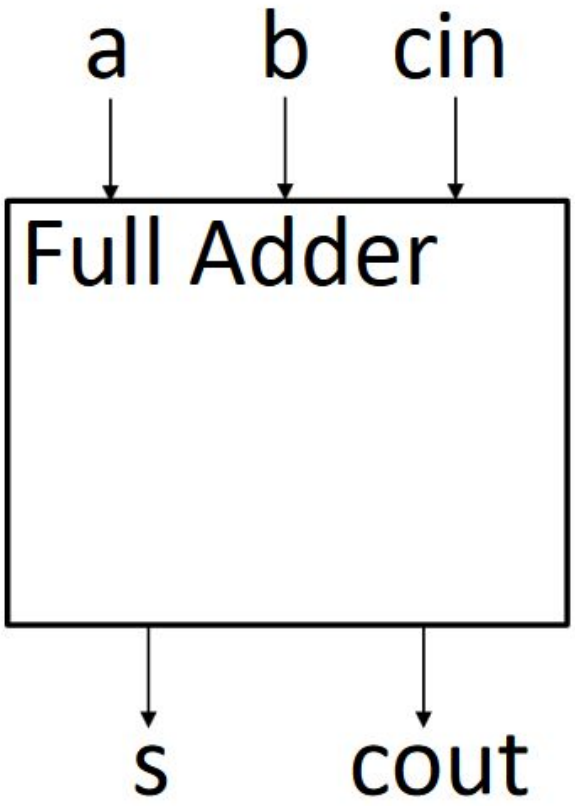
ПЛИС



1. Массив из логических элементов (макроячеек, логических блоков).
2. Блоки входа-выхода (IO).
3. Линии связи между ними и устройство, которое управляет этими связями.

Рис.1 Обобщенная структурная схема ПЛИС.

HDL (Hardware Description Language, рус. Язык описания аппаратуры)
Verilog




```
`timescale 1ns / 1ps
module FullAdder (
        input a,
        input b,
        input cin,
        output s,
        output cout );

      assign {cout,s} = a + b + cin;

endmodule
```

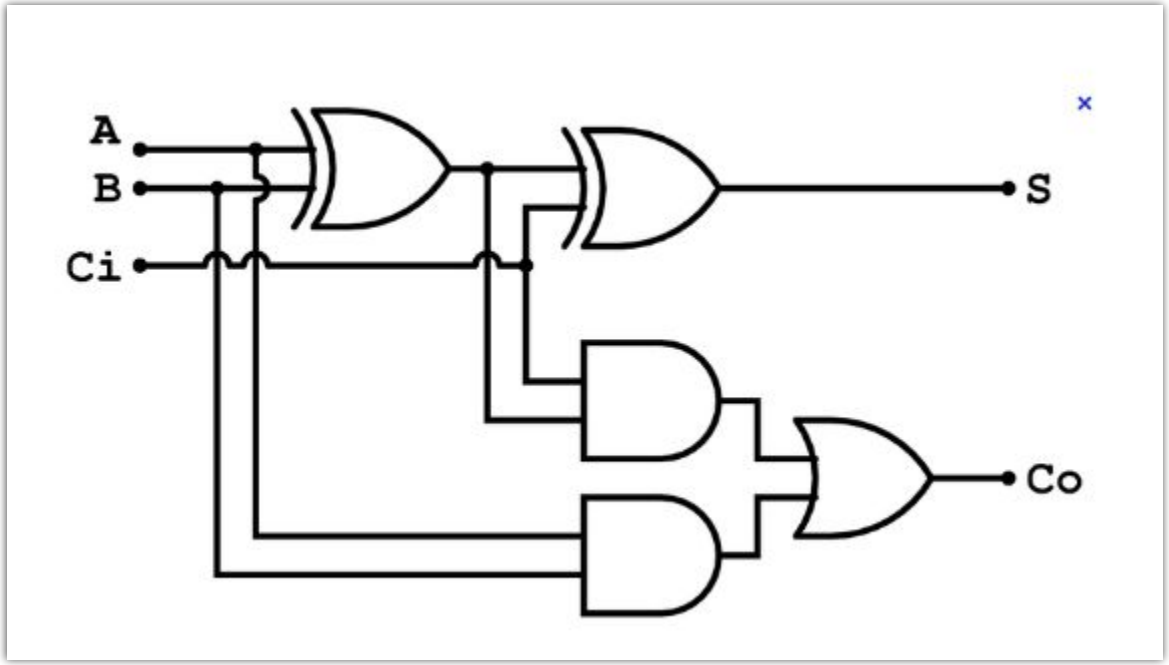
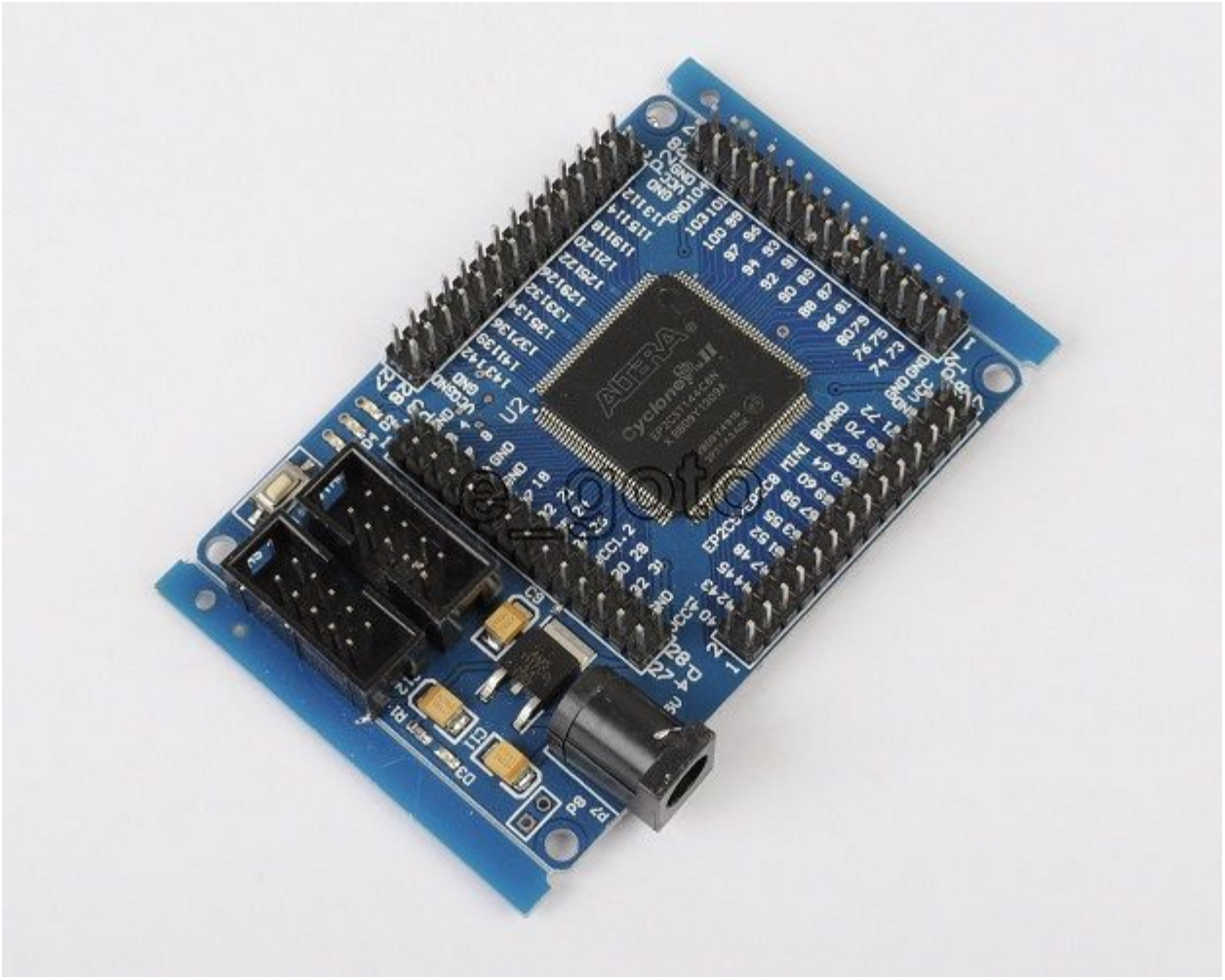

A
B
Ci
S
Co

Таблица 1. Основные характеристики семейства MAX II

Feature	EPM240	EPM570	EPM1270	EPM2210
Логические элементы (LEs)	240	570	1,270	2,210
Эквивалентные макроячейки	192	440	980	1,700
Максимальное количество выводов пользователя	80	160	212	272
Пользовательская Flash память, бит	8,192	8,192	8,192	8,192
t_{PD1} Corner-to-Corner Performance (ns)	4.5	5.5	6.0	6.5
t_{PD2} Fastest Performance (ns)	3.6	3.6	3.6	3.6


ALTERA
Cyclone II
U2

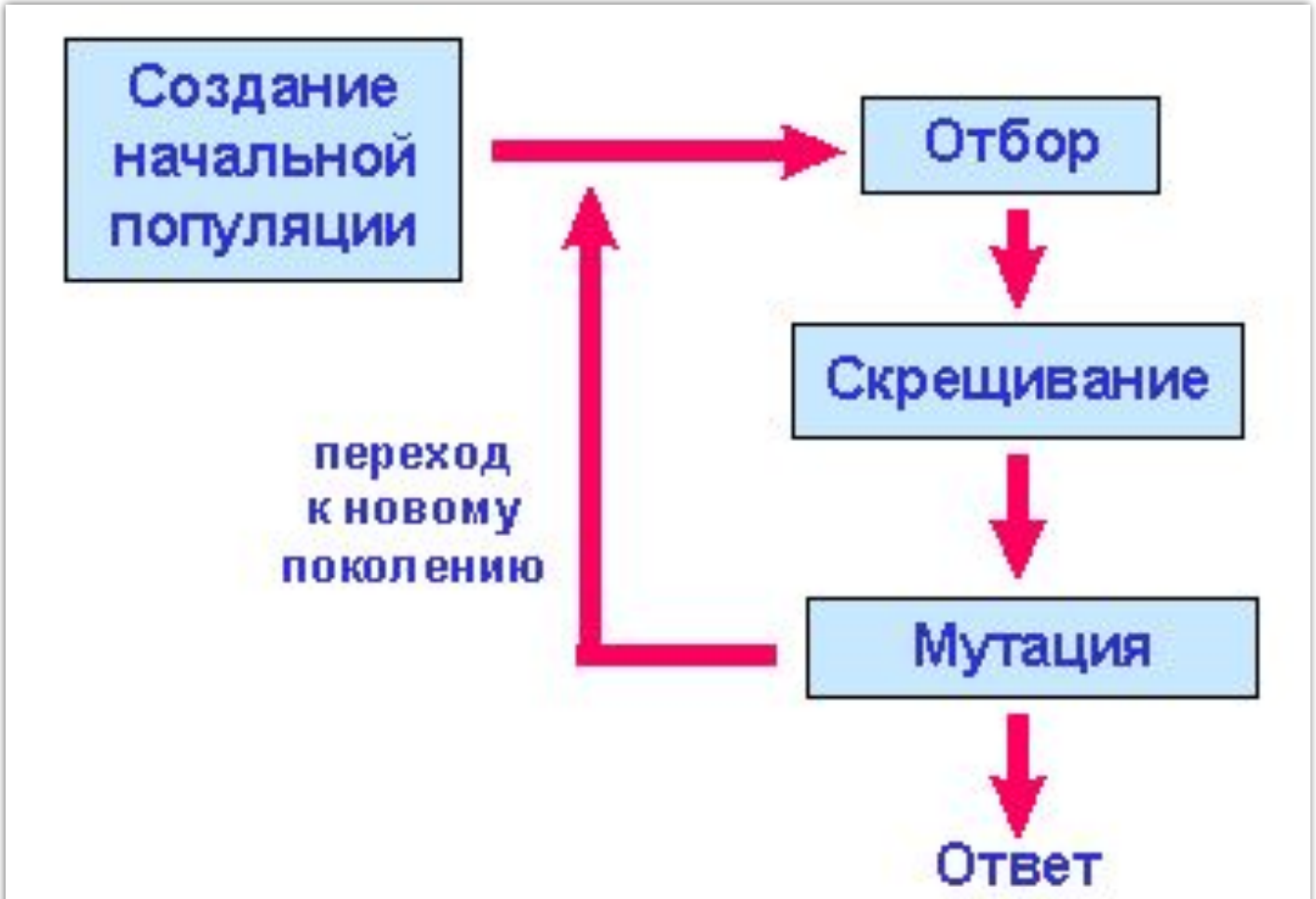
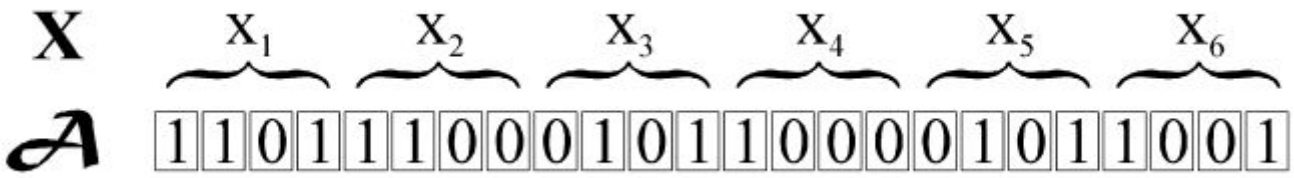
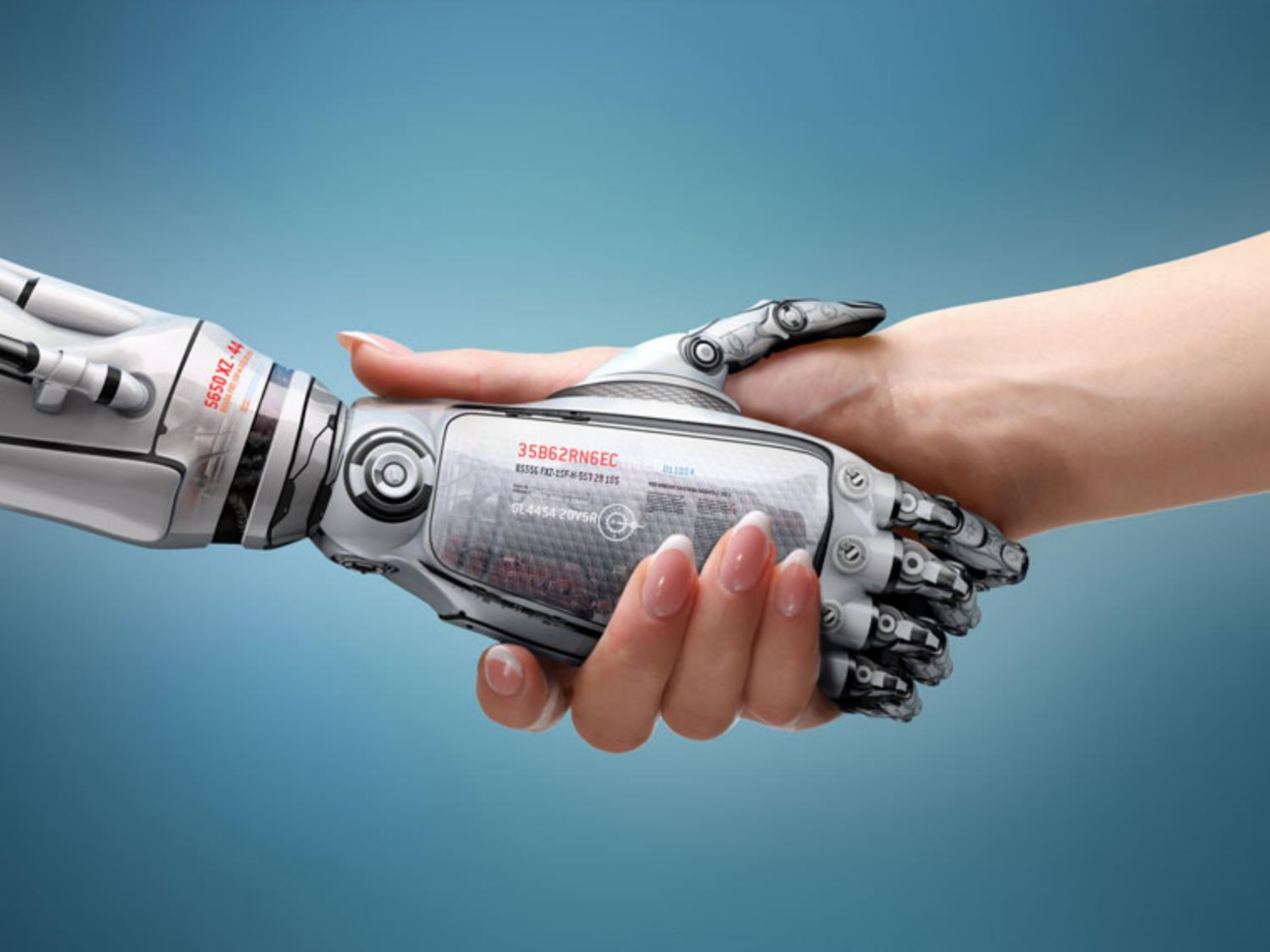


Рис.2. Блок-схема генетического алгоритма.




5650 XL-44
35B62RN6EC

изнес-лидеры в эпоху ИИ
о отличает российских руководителей?[1]

Microso

ВНЕДРЕНИЕ

Россия заняла первое место в мире по активному внедрению

30% российских руководителей активно внедряют искусственный интеллект

В среднем по миру этот показатель равен 22,3%

ОТВЕТСТВЕННОСТЬ

90% из них выразили желание получить поддержку профессионалов

В среднем по миру этот показатель равен 67,3%

30% готовы инвестировать время для адаптации к новым условиям работы

В среднем по миру этот показатель равен 20,3%

ЭТИКА

65% российских директоров считают, что принятие ответственности за этичное применение ИИ – безусловное требование для лидера

В мире такой точки зрения придерживаются 53,9% директоров

БИЗНЕС-ПРИОРИТЕТЫ

Готовы уделять время использованию ИИ

32% для постановки правильных целей

26% для разработки идей

ПОЗИТИВНОЕ ОТНОШЕНИЕ К ИИ

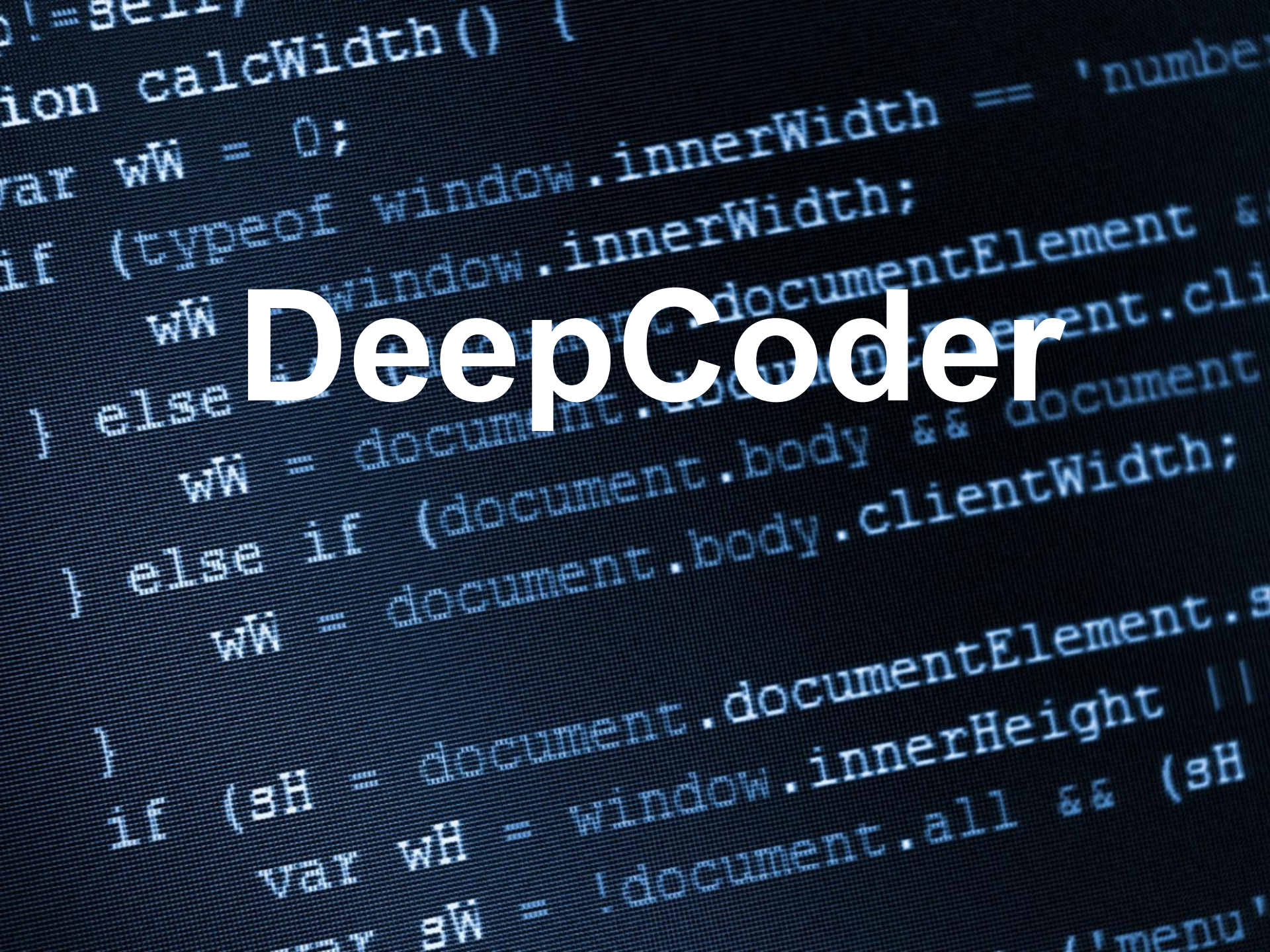
Российские директора заняли второе место по уровню позитивного отношения к ИИ

73% директоров считают, что технология поможет им в их управленческой деятельности

[1] В опросе приняли участие 8 ... уровня высшего звена из ... Германии, Италии, Нид... России, Швейцарии, Велико... США (выборка – 100 респон... стран). Рассматривались ... компаниях со штатом более 25... Исследование проводилось ... 18 по 27 ...

10. Разработчик ПО. Вот уже на протяжении многих лет эта специальность пользуется повышенной популярностью, а освоившие ее люди получают большие зарплаты. Реагируя на потребность в разработчиках ПО, рынок начал предлагать альтернативы — некоторые вендоры стали выпускать инструменты, не требующие специализированных навыков программирования, исследователи Microsoft и Кембриджского университета даже разработали ИИ-инструмент, который может писать код без участия человека. Как и начинающие программисты, он учился создавать код, отталкиваясь от существующих решений.

Пока что этот ИИ не в состоянии создать новую программу с нуля, но есть вероятность, что со временем он сможет сократить отставание от профессионалов. Данные BLS показывают, что разработчики ПО по-прежнему очень востребованная специальность. В прошлом году средняя зарплата программиста составляла 102,3 тыс. долл. Спрос на них вырос на 18%.


DeepCoder