**Лабораторная работа №2**

**Тема**: Графическая поверхность. Эллипс и окружность. Дуга. Сектор. Текст.

**Цель работы:** Получения практических навыков при работе с графическими примитивами.

**Теоретический материал.**

**Эллипс и окружность**

Метод DrawEllipse чертит эллипс внутри прямоугольной области (рис. 2.1). Если прямоугольник является квадратом, то метод рисует окружность.

Инструкция вызова метода DrawEllipse в общем виде выглядит так:

**DrawEllipse(aPen, x, y, w, h);**

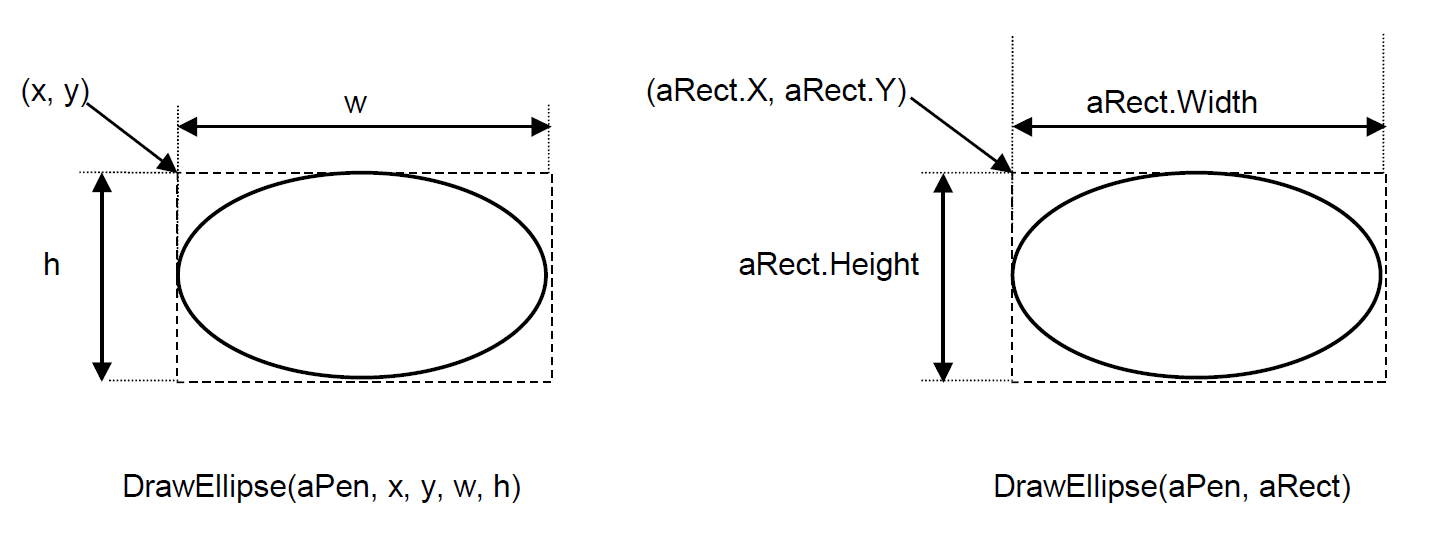


Рис. 2. 1 Метод DrawEllipse рисует эллипс

Параметр aPen, в качестве которого можно использовать один из стандартных карандашей или карандаш, созданный программистом, определяет вид (цвет, толщину, стиль) границы эллипса. Параметры x, y, w и h задают координаты левого верхнего угла и размер прямоугольника, внутри которого метод рисует эллипс.

В инструкции вызова метода DrawEllipse вместо параметров x, y, w и h можно указать структуру типа Rectangle:

**DrawEllipse(aPen, aRect);**

Поля X и Y структуры aRect задают координаты левого верхнего угла области, внутри которой метод рисует эллипс, а поля Width и Height – размер.

Метод FillEllipse рисует закрашенный эллипс. В инструкции вызова метода следует указать кисть (стандартную или созданную программистом), координаты и размер прямоугольника, внутри которого надо нарисовать эллипс:

**FillEllipse(aBrush, x, y, w, h);**

Кисть определяет цвет и способ закраски внутренней области эллипса. Вместо параметров x, y, w и h можно указать структуру типа Rectangle:

**FillEllipse(aBrush, aRect);**

**Дуга**

Метод DrawArc рисует дугу – часть эллипса (рис. 2.2). Инструкция вызова метода в общем виде выглядит так:

**DrawArc(aPen, x, y, w, h, startAngle, sweepAngle)**

Параметры x, y, w и h определяют эллипс (окружность), частью которого является дуга. Параметр startAngle задает начальную точку дуги – пересечение эллипса и прямой, проведенной из центра эллипса и образующей угол startAngle с горизонтальной осью эллипса (угловая координата возрастает по часовой стрелке). Параметр sweepAngle задает длину дуги (в градусах). Если значение sweepAngle положительное, то дуга рисуется от начальной точки по часовой стрелке, если отрицательное – то против. Величины углов задаются в градусах.

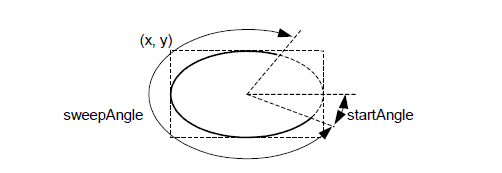


Рис. 2. 2 Значения параметров метода DrawArc определяют дугу как часть эллипса

В инструкции вызова метода DrawArc вместо параметров x, y, w и h можно указать структуру типа Rectangle:

**DrawArc(aPen, aRect, startAngle, sweepAngle)**

**Сектор**

Метод DrawPie рисует границу сектора (рис. 2.3). Инструкция вызова метода выглядит так:

**DrawPie(aPen, x, y, w, h, startAngle, sweepAngle);**

Параметры x, y, w и h определяют эллипс, частью которого является сектор. Параметр startAngle задает начальную точку дуги сектора – пересечение эллипса и прямой, проведенной из центра эллипса и образующей угол startAngle с горизонтальной осью эллипса (угловая координата возрастает по часовой стрелке). Параметр sweepAngle — длину дуги сектора (в градусах). Если значение sweepAngle положительное, то дуга сектора рисуется от начальной точки по часовой стрелке, если отрицательное – против. Величины углов задаются в градусах.

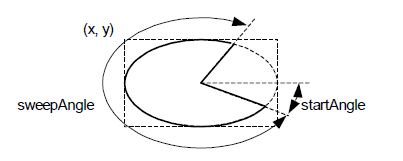


Рис. 2. 3 Значения параметров метода DrawPie определяют сектор как часть эллипса

В инструкции вызова метода DrawPie вместо параметров x, y, w и h можно указать структуру типа Rectangle:

**DrawPie(aPen, aRect, startAngle, sweepAngle)**

Метод FillPie рисует сектор. Параметры у метода FillPie, за исключением первого, вместо которого надо указать кисть, такие же, как и у метода DrawPie.

Программа "Опрос" демонстрирует использование методов DrawPie и FillPie. В ее окне в виде круговой диаграммы отображается результат опроса. Исходные данные для построения диаграммы (вопрос, варианты ответа, количество ответов на каждый из вопросов) загружаются из файла. Окно программы "Опрос" приведено на рис. 2.4, текст — в листинге 2.1.



Рис. 2. 4 Круговая диаграмма в программе "Опрос"

**Листинг 2.1** Круговая диаграмма

**private**:

String^ header; *// заголовок диаграммы*

*// количество элементов данных*

**int** N;

**array**<**double**>^ dat; *// ряд данных*

**array**<**double**>^ p; *// доля категории (ряда) в общей сумме*

*// подписи данных*

**array**<String^>^ title;

*// конструктор*

Form1(**void**) {

InitializeComponent();

System::IO::StreamReader^ sr;

**try** {

sr = **gcnew** System::IO::StreamReader(

Application::StartupPath + "\\date.dat",

System::Text::Encoding::GetEncoding(1251));

*// считываем заголовок диаграммы*

header = sr->ReadLine();

*// считываем данные о количестве записей*

N = Convert::ToInt16(sr->ReadLine());

*// и инициализируем массивы*

dat = **gcnew array**<**double**>(N);

p = **gcnew array**<**double**>(N);

title = **gcnew array**<String^>(N);

*// читаем данные*

**int** i = 0;

String^ st;

st = sr->ReadLine();

**while** ((st != String::Empty) && (i < N)){

title[i] = st;

st = sr->ReadLine();

dat[i++] = Convert::ToDouble(st);

st = sr->ReadLine();

}

*// закрываем поток*

sr->Close();

*// Данные загружены.*

*// Задаем функцию обработки события Paint*

**this**->Paint += **gcnew** System::Windows::

Forms::PaintEventHandler (**this**, &Form1::drawDiagram);

**double** sum = 0;

**int** j = 0;

*// вычислить сумму*

**for** (j = 0; j < N; j++)

sum += dat[j];

*// вычислить долю каждой категории*

**for** (j = 0; j < N; j++)

p[j] = (**double**)(dat[j] / sum);

}

**catch** (System::IO::FileNotFoundException^ ex) {

MessageBox::Show(ex->Message, "Диаграмма",

MessageBoxButtons::OK, MessageBoxIcon::Error);

}

}

}

*// рисуем круговую диаграмму*

**void** drawDiagram(System::Object^ sender, System::Windows::Forms::PaintEventArgs^ e) {

*// графическая поверхность*

System::Drawing::Graphics^ g = e->Graphics;

*// шрифт заголовка*

System::Drawing::Font^ hFont = **gcnew** System::Drawing::Font("Tahoma", 12);

*// выводим заголовок*

**int** w = (**int**)g->MeasureString(header, hFont).Width;

**int** x = (ClientSize.Width — w) / 2;

g->DrawString(header,hFont, Brushes::Black, x, 10);

*// шрифт легенды*

System::Drawing::Font^ lFont = **gcnew** System::Drawing::Font("Tahoma", 9);

*// диаметр диаграммы*

**int** d = ClientSize.Height — 80;

**int** x0 = 30;

**int** y0 = (ClientSize.Height — d) / 2 + 10;

*// координаты левого верхнего угла области легенды*

**int** lx = 60 + d;

**int** ly = y0 + (d — N \* 20 + 10) / 2;

**int** swe; *// длина дуги сектора*

*// кисть для заливки сектора диаграммы*

System::Drawing::Brush^ fBrush = Brushes::White;

*// начальная точка дуги сектора*

**int** sta = -90;

*// рисуем диаграмму*

**for** (**int** i = 0; i < N; i++) {

*// длина дуги*

swe = (**int**)(360 \* p[i]);

*// задать цвет сектора*

**switch** (i) {

**case** 0: fBrush = Brushes::Gold; **break**;

**case** 1: fBrush = Brushes::Silver; **break**;

**case** 2: fBrush = Brushes::DarkGoldenrod; **break**;

**case** 3: fBrush = Brushes::Azure; **break**;

**case** 4: fBrush = Brushes::OrangeRed; **break**;

**case** 5: fBrush = Brushes::RoyalBlue; **break**;

**case** 6: fBrush = Brushes::SteelBlue; **break**;

**case** 7: fBrush = Brushes::Chocolate; **break**;

**case** 8: fBrush = Brushes::LightGray; **break**;

}

*// Из-за округления возможна ситуация, при которой*

*// будет промежуток между последним и первым секторами*

**if** (i == N — 1)

*// последний сектор*

swe = 270 — sta;

*// рисуем сектор*

g->FillPie(fBrush, x0, y0, d, d, sta, swe);

*// рисуем границу сектора*

g->DrawPie(Pens::Black, x0, y0, d, d, sta, swe);

*// прямоугольник легенды*

g->FillRectangle(fBrush, lx, ly + i \* 20, 20, 10);

g->DrawRectangle(Pens::Black, lx, ly + i \* 20, 20, 10);

*// подпись*

g->DrawString(title[i] + " — " +

p[i].ToString("P"),

lFont, Brushes::Black, lx + 24, ly + i \* 20 — 3);

*// начальная точка дуги для следующего сектора*

sta = sta + swe;

}

}

**private**: System::Void Form1\_Resize(System::Object^ sender, System::EventArgs^ e) {

**this**->Refresh();

}

**Текст**

Вывод текста на графическую поверхность выполняет метод DrawString. В инструкции вызова метода указывается строка, шрифт, кисть и координаты точки, от которой надо вывести текст:

**DrawString(st, aFont, aBrush, x, y);**

Параметр st задает текст, параметр aFont – шрифт, который используется для отображения текста, a aBrush — цвет текста. Параметры x и y определяют координаты левого верхнего угла области отображения текста (рис. 2.5).



Рис. 2. 5 Координаты и размер области отображения текста

В программе, окно которой показано, текст выводит функция обработки события Paint.

**private**: System::Void Form1\_Paint(System::Object^ sender,

System::Windows::Forms::PaintEventArgs^ e){

String^ st1 = "Microsoft Visual Studio 2010";

e->Graphics->DrawString(st1,**this**->Font, Brushes::Black, 20,20);

}

В приведенном примере для вывода текста используется шрифт формы, заданный свойством Font. Если текст надо вывести шрифтом, отличным от шрифта, заданного для формы, то этот шрифт следует создать – объявить и инициализировать объект типа Font.

Инструкция создания шрифта (вызова конструктора) выглядит так:

System::Drawing::Font^ aFont = **gcnew** System::Drawing::Font(FontFamily,Size,FontStyle);

Параметр FontFamily (строкового типа) задает шрифт, на основе которого создается новый (определяет семейство, к которому относится создаваемый шрифт). В качестве значения параметра FontFamily можно использовать название шрифта, зарегистрированного в системе (Arial, Times New Roman, Tahoma). Параметр Size задает размер (в пунктах) шрифта. Параметр FontStyle определяет стиль символов шрифта (FontStyle::Bold — полужир-ный; FontStyle::Italic — курсив; FontStyle::UnderLine — подчеркнутый). Параметр FontStyle можно не указывать. В этом случае будет создан шрифт обычного начертания (FontStyle::Regular).

Следует обратить внимание, что изменить характеристики созданного шрифта нельзя (свойства FontFamily, Size и Style объекта Font определены "только для чтения"). Поэтому если в программе предполагается использовать разные шрифты, их необходимо создать.

В листинге 2.2 приведена функция обработки события Paint, демонстрирующая создание и использование шрифтов для вывода текста на поверхность формы (рис. 2.6).



Рис. 2. 6 Вывод текста на поверхность формы методом DrawString

**Листинг 2.2 Создание и использование шрифта**

**private**: System::Void Form1\_Paint(System::Object^ sender,

System::Windows::Forms::PaintEventArgs^ e) {

**int** x,y;

x = 10;

y = 10;

String^ st = "Microsoft Visual Studio 2010";

System::Drawing::Font^ rFont = **gcnew** System::Drawing::Font

("Tahoma", 11, FontStyle::Regular);

System::Drawing::Font^ bFont = **gcnew** System::Drawing::Font

("Tahoma", 11, FontStyle::Bold);

System::Drawing::Font^ iFont = **gcnew** System::Drawing::Font

("Tahoma", 11, FontStyle::Italic);

e->Graphics->DrawString(st, rFont, Brushes::Black, x, y);

e->Graphics->DrawString(st, bFont, Brushes::Black, x, y+20);

e->Graphics->DrawString(st, iFont, Brushes::Black, x, y+40);

}

Метод DrawString позволяет вывести текст в прямоугольную область. Причем, если длина текста такая, что вывести его в одну строку нельзя, он будет выведен в несколько строк.

Инструкция вызова метода DrawString, обеспечивающая вывод текста в область, выглядит так:

**DrawString(st, aFont, aBrush, aRec);**

Параметр aRec задает положение и размер области вывода текста.

В листинге 2.3 приведена функция обработки события Paint, которая де-монстрирует вывод текста в область. Окно программы приведено на рис. 2.7 (границы области вывода текста показаны для наглядности).

Листинг 2.3. **Вывод текста в область**

**private**: System::Void Form1\_Paint(System::Object^ sender,

System::Windows::Forms::PaintEventArgs^ e) {

String^ st1 = "У лукоморья дуб зеленый;\nЗлатая цепь на дубе том:\n"

"И днем и ночью кот ученый\nВсе ходит по цепи кругом;";

*// положение и размер области вывода текста*

Rectangle aRect = Rectangle(10,10,200,90);

*// вывести текст*

e->Graphics->DrawString(st1, **this**->Font, Brushes::Black, aRect);

*// показать область отображения текста*

e->Graphics->DrawRectangle(Pens::Gray, aRect);

}



Рис. 2. 7 Метод DrawString позволяет вывести текст в область

Часто надо знать, сколько места займет текст, например, для того, чтобы правильно разместить его на графической поверхности. Так, чтобы разместить текст по центру формы, надо знать ширину области вывода. Очевидно, что размер области вывода зависит от шрифта, который используется для отображения текста.

Информацию о размере области вывода текста возвращает метод MeasureString. В инструкции вызова метода надо указать строку и шрифт, который будет использован для ее отображения. Значением метода MeasureString является объект типа SizeF, свойства Width и Height которого содержат информацию о размере области вывода текста.

В листинге 2.4 приведен фрагмент программы, которая демонстрирует использование метода MeasureString. Функция обработки события Paint выводит текст в центре формы (рис. 2.8). Кроме обработки события Paint про-грамма выполняет обработку события Resize, которое возникнет, если поль-зователь изменит размер окна. Обработка заключается в вызове метода Refresh, который информирует систему о необходимости обновить (перери-совать) окно, что приводит к возникновению события Paint.

Листинг 2.4 **Использование метода MeasureString**

**private**: System::Void Form1\_Paint(System::Object^ sender,

System::Windows::Forms::PaintEventArgs^ e) {

String^ st1 = "Microsoft Visual Studio 2010";

System::Drawing::Font^ h1 = **gcnew** System::Drawing::Font("Tahoma",12);

System::Drawing::Font^ h2 = **gcnew** System::Drawing::Font("Tahoma",10);

System::Drawing::Font^ h3 = **gcnew** System::Drawing::Font("Tahoma",9);

float x,y;

float w,h;

w = e->Graphics->MeasureString(st1, h1).Width;

h = e->Graphics->MeasureString(st1, h1).Height;

x = (**this**->ClientSize.Width — w)/2;

y = 10;

e->Graphics->DrawString(st1, h1, Brushes::Black, x, y);

w = e->Graphics->MeasureString(st1, h2).Width;

h = e->Graphics->MeasureString(st1, h2).Height;

x = (**this**->ClientSize.Width — w)/2;

y += h;

e->Graphics->DrawString(st1,h2, Brushes::Black, x, y);

w = e->Graphics->MeasureString(st1, h3).Width;

h = e->Graphics->MeasureString(st1, h3).Height;

x = (**this**->ClientSize.Width — w)/2;

y += h;

e->Graphics->DrawString(st1,h3, Brushes::Black, x, y);

}

*// изменился размер формы*

**private**: System::Void Form1\_Resize(System::Object^ sender, System::EventArgs^ e){

*// Сообщить системе о необходимости перерисовать окно.*

*// В результате будет сгенерировано событие Paint.*

**this**->Refresh();

}



Рис. 2. 8 Метод MeasureString позволяет получить информацию о размере области вывода текста, чтобы расположить текст по центру формы

**Задания.**

1. Построить рисунок, соответствующую Вашему варианту. Рисунки должны быть цветными:

|  |  |
| --- | --- |
| Вариант | Рисунок |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  | Дорисуйте недостающие элементы. |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

1. Построить секторную диаграмму с легендой соответствующую Вашему варианту:

|  |  |
| --- | --- |
| Вариант | Данные |
|  | |  |  |  |  | | --- | --- | --- | --- | | Успеваемость | отличники | хорошисты | троечники | | Кол-во студентов | 5 | 12 | 6 | |
|  | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Год | 2010 | 2011 | 2012 | 2013 | | Прибыль | 20000 | 15000 | 21000 | 22000 | |
|  | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Продукт | помидоры | лук | морковь | огурцы | | Цена | 3,58 | 3,99 | 1,65 | 7,25 | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | Страна | США | Англия | Болгария | Турция | Черногория | Греция | | Кол-во туристов | 100 | 150 | 125 | 200 | 180 | 215 | |
|  | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | Продукты | мука | сахар | яйца | масло | яблоки | мед | | Вес | 450 | 200 | 180 | 200 | 400 | 300 | |
|  | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | Этажность домов | 3 | 5 | 9 | 12 | 14 | 16 | | Кол-во | 10 | 15 | 18 | 5 | 5 | 3 | |
|  | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Успеваемость | отличники | хорошисты | троечники | неуспевающие | | Кол-во студентов | 5 | 12 | 6 | 2 | |
|  | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Месяц | май | июнь | июль | август | | Кол-во перевозок | 500 | 550 | 600 | 750 | |
|  | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | Страна | Польша | Венгрия | Болгария | Чехия | Словения | Румыния | | Кол-во представителей | 12 | 9 | 13 | 20 | 12 | 10 | |
|  | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | Продукты | мука | сахар | яйца | масло | яблоки | мед | | Вес | 0,5 | 0,2 | 1,8 | 0,2 | 0,5 | 0,25 | |
|  | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Групп | младшая | средняя | старшая | подготовительная | | Кол-во студентов | 10 | 18 | 20 | 25 | |
|  | |  |  |  |  |  | | --- | --- | --- | --- | --- | | кол-во проживающих | 1 | 2 | 3 | 4 | | кол-во номеров | 5 | 10 | 10 | 8 | |
|  | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Фирма | Свиточ | Рошен | АВК | Стимул | | прибыль | 500 | 900 | 750 | 600 | |
|  | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | Год | 2010 | 2011 | 2012 | 2013 | 2014 | | Затраты | 20 | 35 | 12 | 10 | 10 | |
|  | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Продукт | яблоки | груши | малина | виноград | | Цена | 10 | 15 | 30 | 30 | |
|  | |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | Язык | Английский | Французский | Немецкий | Китайский | Арабский | | Кол-во обучающихся | 30 | 25 | 30 | 40 | 10 | |
|  | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | консервация | помидоры | помидоры (сок) | огурцы | лечо | икра | соус | | Кол-во банок | 30 | 20 | 30 | 10 | 5 | 15 | |
|  | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | Этажность домов | 3 | 5 | 9 | 12 | 14 | 16 | | Кол-во | 5 | 10 | 20 | 7 | 7 | 7 | |
|  | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Успеваемость | отличники | хорошисты | троечники | неуспевающие | | Кол-во студентов | 7 | 10 | 7 | 1 | |
|  | |  |  |  |  |  | | --- | --- | --- | --- | --- | | Квартал | 1 | 2 | 3 | 4 | | Кол-во перевозок | 1500 | 2000 | 2000 | 1600 | |
|  | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | Страна | Польша | Венгрия | Болгария | Чехия | Словения | Румыния | | Кол-во команд | 5 | 4 | 6 | 4 | 4 | 2 | |
|  | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | Продукты | мука | сахар | яйца | масло | яблоки | мед | | Вес | 0,5 | 0,2 | 1,8 | 0,2 | 0,5 | 0,25 | |