**Рекомендации №2:**

1. Рассчитываю на то, что ПРИМЕР табулирования функции вы сделали..

Поэтому продолжаем работать с ним.

Теперь эту же задачу решите, формируя динамический массив.

В **заголовочном файле диалога** он уже будет объявлен, например :

int m = 11;

int n = 2;

double\*\* arr;

И добавим следом в заголовочный файл объявление функций выделения и освобождения пвмяти…

void allocMem(int m, int n, double \*\*&arr);

void freeMem(int m, int n, double\*\*&arr);

Коды этих функций, конечно, должны быть включены в исходный файл диалога:

Выделение памяти:

void CMFCFile1Dlg::allocMem(int m, int n, double \*\*&arr)

{

int i;

arr = new double\* [m];

for (i = 0; i < m; i++)

{

arr [i] = new double[n];

}

}

Высвобождение памяти:

void CMFCFile1Dlg::freeMem(int m, int n, double\*\*&arr)

{

int i;

for (i = 0; i < m; i++)

{

delete[] arr [i];

}

delete[] arr;

}

Вспомнили?....

Теперь подумайте, где вы их примените.

Добавьте на диалоговом окне **кнопку ВЫХОД** и создайте для нее обработчик.

void CMFCFile1Dlg::OnBnClickedButtonExit()

{

// TODO: добавьте свой код обработчика уведомлений

**freeMem(m, n, arr);**

OnOK();

}

Вызов функции allocMem(m, n, arr); нужен в обработчике, где происходит формирование массива, согласны?....

void CMFCFile1Dlg::OnBnClickedCheck()

{

// TODO: добавьте свой код обработчика уведомлений

UpdateData(TRUE);

**allocMem(m, n, arr);**

dx = (m\_EditB - m\_EditA) / 10;

CString Str;

if (m\_Check.GetCheck() == TRUE)

{

for (int i = 0; i <= 10; i++)

{

arr[i][0] = m\_EditA + i \* dx;

arr[i][1] = arr[i][0] \* arr[i][0];

Str.Format(L"%f \n %f", arr[i][0], arr[i][1]);

m\_List.AddString(Str);

}

}

else

{

m\_EditA = 0.;

m\_EditB = 0.;

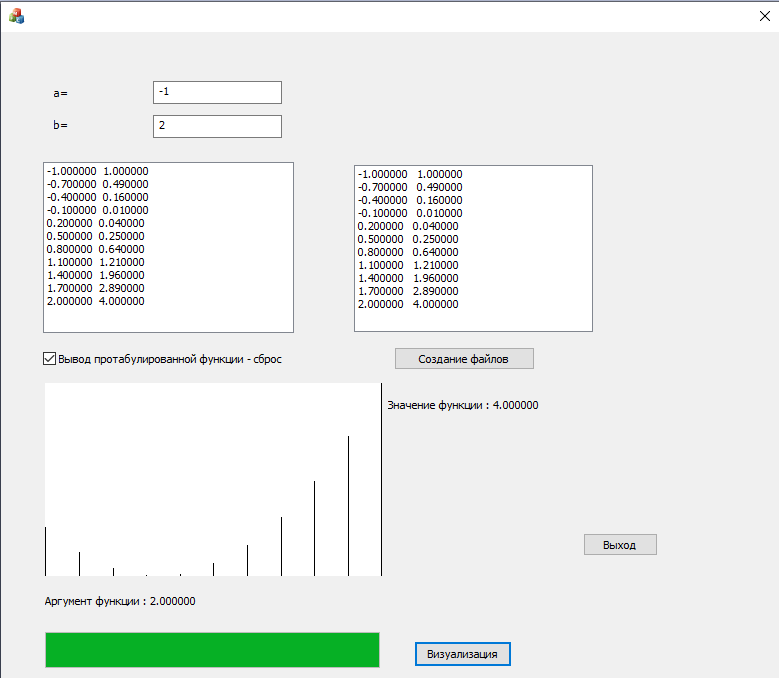
UpdateData(FALSE);

m\_List.ResetContent();

}

}

1. Отображение результатов табулирования функции графически.



* 1. Добавьте элемент управления Picture Control.
  2. Progress bar (Линейный индикатор) должен изменяться синхронно с графическим табулированием функции .

m\_ProgrCalk.SetPos(z);

Аргумент функции SetPos(z) должен быть **целым** числом! Поэтому задавать напрямую значение arr[i][0] может вызыать ошибку

* 1. Значения аргумента и функции отражаются в StaticText рядом с осями.Изменяются тоже синхронно с графическим табулированием.

Можно сделать так:

CString StrX;

StrX.Format(L"Аргумент функции : %f", arr[i][0]);

SetDlgItemTextW(IDC\_STATIC\_X, StrX);

CString StrY;

StrY.Format(L"Значение функции : %f", arr[i][1]);

SetDlgItemTextW(IDC\_STATIC\_Y, StrY);

UpdateData(FALSE);

* 1. График не должен выходить за пределы окна Picture Control. Должен быть предусмотрен масштаб.

// Расчитываем масштабный коэфициент по оси х

double dScaleX = (rect.right - rect.left) / 10.;

Изменение масштаба по оси Y dScaleY подумайте сами….

2.5 Для рисования графика используются две функции класса CClientDC:

Первая перемещает курсор в заданные координаты. Вторая рисует линию из текущего положения курсора в заданные координаты.

Функция ceil() округляет аргумент до ближайшего целого снизу. Это необходимо делать потому, что аргументы функций MoveTo() и LineTo() должны быть целочисленными.

dc.MoveTo(ceil(i \* dScaleX), rect.bottom-1 );

dc.LineTo(ceil(i \* dScaleX), rect.bottom-1 - ceil( arr[i][1] \* dScaleY));

Sleep(300); // Задержка по времени

* 1. Обработчик для кнопки ВИЗУАЛИЗАЦИЯ запускает весь этот процесс.

1. Не забудьте убрать сортировку в обоих ListBox. Это устнавливается в **Свойствах** элемента управления. Иначе, если зададите интервал, который включает в свой диапазон и отрицательные числа, результат табуляции будет отсортирован….Поэкспериментируйте…☺