Пример использования подпрограмм *x*GEMM из библиотеки BLAS в программах на языке Си

BLAS (Basic Linear Algebra Subroutines) – библиотека базовых подпрограмм линейной алгебры. Это набор подпрограмм для выполнения базовых операций над векторами и матрицами. Библиотека BLAS имеет стандартные интерфейсы для языков Fortran и C.

Процедура *x*GEMM выполняет следующую операцию: $C = \alpha \cdot op(A) \cdot op(B) + \beta \cdot C$

где: A – матрица M×K,
B – матрица K×N,
C – матрица M×N,
α, β – коэффициенты,
op( ) – некоторая операция над матрицами.

Первая буква в названии процедуры определяет тип данных элементов матрицы:

S – вещественный с одинарной точностью (например, SGEMM),
D – вещественный с двойной точностью (например, DGEMM),
C – комплексный с одинарной точностью (например, CGEMM),
Z – комплексный с двойной точностью (например, ZGEMM).

Прототип функции SGEMM в интерфейсе для языка C выглядит следующим образом:

```
void cblas_sgemm(const enum CBLAS_ORDER Order,
   const enum CBLAS_TRANSPOSE TransA, const enum CBLAS_TRANSPOSE TransB,
   const int M, const int N,
   const int K, const float alpha, const float *A,
   const int lda, const float *B, const int ldb,
   const float beta, float *C, const int ldc);
```

где: Order – определяет способ хранения матриц в памяти:
CblasRowMajor – матрицы хранятся по строкам (стандартно в C),
CblasColMajor – матрицы хранятся по столбцам (стандартно в Fortran);

TransA, TransB – определяют операции op() над матрицами A и B:
CblasNoTrans – ничего не делать,
CblasTrans – выполнить транспонирование матрицы,
CblasConjTrans – вычислить сопряженную матрицу;

M, N, K – размеры матриц;

alpha, beta – коэффициенты;

lda, ldb, ldc – число элементов в ведущей размерности матрицы (строке или столбце). Для массивов в языке Си – это число элементов в строке соответствующей матрицы: lda = K, ldb = N, ldc = N.

Для использования процедур из библиотеки BLAS, необходимо, чтобы массивы в памяти хранились одним непрерывным блоком. Например, массив A из следующего примера нельзя передавать в качестве параметра в BLAS-процедуру, т.к. в нем не гарантировано, что строки матрицы будут следовать в памяти одна за другой:

```
float **A=(float**)malloc(M*sizeof(float*));
for (i=0;i<M;i++) A[i]=(float*)malloc(K*sizeof(float));
```

Если необходимо обращаться к динамическому массиву как к двумерному, то можно поступить следующим образом:

```
float **B=(float**)malloc(M*sizeof(float*));
float *B1=(float*)malloc(M*K*sizeof(float)); // непрерывный
for (i=0;i<M;i++) B[i]=&B1[i*K];
```

Пример программы умножения двух матриц с использованием BLAS:

```
#include<stdio.h>
#ifdef __INTEL_COMPILER
  #include<mkl_cblas.h>  // C-интерфейс BLAS (заголовочный файл из MKL)
#else
  #include<cblas.h>  // C-интерфейс BLAS (стандартный заголовочный файл)
#endif

#define M 300
#define N 400
#define K 500

float A[M*K],    // массив расположен в памяти одним непрерывным блоком
      B[K][N],   // этот тоже
      *C;        // и этот тоже будет непрерывным
int main()
{ int i,j;
  C=(float*)malloc(M*N*sizeof(float)); // выделение непрерывного блока
  for (i=0;i<M;i++)      // инициализация массива A
    for (j=0;j<K;j++)
      A[i*K+j]=3*j+2*i;
  for (i=0;i<K;i++)      // инициализация массива B
    for (j=0;j<N;j++)
      B[i][j]=5*i+j;
  for (i=0;i<M*N;i++) C[i]=5; // инициализация массива C (зачем-то)
  // перемножение матриц
  cblas_sgemm(CblasRowMajor,CblasNoTrans,CblasNoTrans,
              M,N,K,1.0,A,K,&B[0][0],N,0.0,C,N);
  for (i=0;i<M;i++) // выводим результат на экран
  { for (j=0;j<N;j++) printf("%.2f ",C[i*N+j]);
    printf("\n");
  }
  free(C);
  return 0;
}
```

Пример команд компиляции:

- С использованием компилятора gcc и библиотеки ATLAS:
  ```
  gcc -I/usr/local/atlas/include -L/usr/local/atlas/lib \
      -O3 -o prog_gcc_atlas prog.c -lcblas -latlas
  ```

- С использованием компилятора от Intel и библиотеки MKL:
  ```
  icc -mkl -O3 -o prog_icc_mkl prog.c
  ```