Основы C/C++ – Домашнее задание 15

Пеганов Антон
peganoff2@mail.ru

6 апреля 2020 г.

Задача 1

Напишите функцию `add_matrices()`, выполняющую сложение матриц.

Задача 2

Напишите функцию `scalar_mul()`, выполняющую умножение матрицы на число.

Задача 3

Напишите функцию `matmul()`, выполняющую умножение матриц.

Адресная арифметика

Адресная арифметика — выполнение сложениия и вычитания с указателями.

Общая информация об адресах и указателях

Оперативная память компьютера поделена на байты (1 байт = 8 бит). Переменная занимает 1 или более байт, расположенных подряд. Например, переменная типа `int` занимает 4 байта, а переменная типа `char` — 1 байт. Байты нумеруются слева направо. Номер первого байта, занимаемого в переменной, и есть адрес переменной в памяти.

Не думайте, что адрес переменной, который возвращает оператор `&`, — это не номер байта в оперативной памяти в бытовом понимании. На самом деле это виртуальный адрес, который затем преобразуется операционной системой в физический адрес.

Если распечатать адрес, то он будет будет представлен в 16-тиричной системе счисления. Например, `0x7ffefbc8abe4` (пара символов `0x` не является частью числа, а только показывает, что число записано в 16-тиричной системе счисления). Количество памяти, занимаемое указателем зависит от многих факторов: разрядности процессора, настроек компилятора, типа данных, на который ссылается указатель.

Сложение адреса и целого числа

Пусть указатель `type *p` складывается с целым числом `i`. При выполнении этой операции оперативную память можно воспринимать, как массив `type M[]` переменных типа `type`. Сложение возвращает указатель на элемент `M`, расположенный на `i` позиций правее элемента, на который указывает `p`. Разумеется, если `i < 0`, то `p+i` будет указывать на элемент расположенный на `i` позиций левее.

Важно понимать, что результат сложения указателя и целого числа зависит от типа, на который указывает указатель. Например, если это указатель на целое число, то прибавление единицы указателю увеличивает его на 4. Это связано, что единицей измерения памяти является байт, а тип `int` занимает 4 байта памяти. Таким образом, если `p` — указатель на 'long long', то `p+1` будет отличаться от `p` на 8, а если `p` — указатель на `char`, то `p+1` будет больше `p` на 1.

См. пример *cpp/hw15/address_ adding.cpp* в нашем репозитории.

Вычитание указателей

Разность двух указателей на переменные типа `type` показывает, сколько переменных типа `type` укладывается между местами в памяти, на которые ссылаются указатели. Например, если два указателя на переменные типа `char` равны `p1 = 0x3a` и `p2 = 0x3f`, то `p2 - p1 == 5`. Вычитание указателей — операция, обратная сложению указателя и целого числа, и поэтому возвращает целое число.

См. пример *cpp/hw15/subtracting_ addresses.cpp* в нашем репозитории.

Задача 4

Создайте двух мерный динамический массив `int **arr` и распечатайте:

1) адрес указатель на массив `arr`,
2) разность адресов `arr[0] - arr`,
3) разности адресов `arr[i+1] - arr[i]`.

Как объясняются полученные результаты? Какие проблемы Вы видите в такой реализации динамического массива?