Смирнов Иван ИУ7-22Б - 2023г.

Отчет

Задание №3.2

Отладка

Целью работы является изучение представления многомерного статического массива в памяти.

1.

В программе задан трехмерный массив целых чисел, размеры которого равны 2, 3, 4 соответсвенно.

…

```
#define I 2

#define J 3

#define K 4
```

…

```
int a[I][J][K] = {{{1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12}}, {{13, 14, 15, 16}, {17, 18, 19, 20}, {21, 22, 23, 24}}};
```

…

2.

С помощью команды отладчика gdb “x /nfu”, узнаем дамп памяти данного трехмерного массива. Но для начала узнаем размер массива в байтах.

```
(gdb) print sizeof(a)

$6 = 96

(gdb) x /96xb a

0x7fffffffdec0:    0x01  0x00  0x00  0x00  0x02  0x00  0x00  0x00

0x7fffffffdec8:    0x03  0x00  0x00  0x00  0x04  0x00  0x00  0x00

0x7fffffffded0:    0x05  0x00  0x00  0x00  0x06  0x00  0x00  0x00

0x7fffffffded8:    0x07  0x00  0x00  0x00  0x08  0x00  0x00  0x00

0x7fffffffdee0:    0x09  0x00  0x00  0x00  0x0a  0x00  0x00  0x00

0x7fffffffdee8:    0x0b  0x00  0x00  0x00  0x0c  0x00  0x00  0x00
```

```
0x7fffffffdef0:    0x0d  0x00  0x00  0x00  0x0e  0x00  0x00  0x00
0x7fffffffdef8:    0x0f  0x00  0x00  0x00  0x10  0x00  0x00  0x00
0x7fffffffdf00:    0x11  0x00  0x00  0x00  0x12  0x00  0x00  0x00
0x7fffffffdf08:    0x13  0x00  0x00  0x00  0x14  0x00  0x00  0x00
0x7fffffffdf10:    0x15  0x00  0x00  0x00  0x16  0x00  0x00  0x00
0x7fffffffdf18:    0x17  0x00  0x00  0x00  0x18  0x00  0x00  0x00
```

3.

С помощью той же команды можно увидеть дамп памяти каждого из компонентов массива.

```
(gdb) print a
$10 = {{{1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12}}, {{13, 14, 15,
16}, {17, 18, 19, 20}, {21, 22, 23, 24}}}
(gdb) print a[0]
$11 = {{1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12}}
(gdb) print a[0][0]
$12 = {1, 2, 3, 4}
(gdb) print a[0][0][0]
$13 = 1
(gdb) x /48xb a[0]
0x7fffffffdec0:    0x01  0x00  0x00  0x00  0x02  0x00  0x00  0x00
0x7fffffffdec8:    0x03  0x00  0x00  0x00  0x04  0x00  0x00  0x00
0x7fffffffded0:    0x05  0x00  0x00  0x00  0x06  0x00  0x00  0x00
0x7fffffffded8:    0x07  0x00  0x00  0x00  0x08  0x00  0x00  0x00
0x7fffffffdee0:    0x09  0x00  0x00  0x00  0x0a  0x00  0x00  0x00
0x7fffffffdee8:    0x0b  0x00  0x00  0x00  0x0c  0x00  0x00  0x00
(gdb) x /16xb a[0][0]
0x7fffffffdec0:    0x01  0x00  0x00  0x00  0x02  0x00  0x00  0x00
0x7fffffffdec8:    0x03  0x00  0x00  0x00  0x04  0x00  0x00  0x00
(gdb) x /4xb &a[0][0][0]
0x7fffffffdec0:    0x01  0x00  0x00  0x00
```

4.

Компонент	Указатель	Размер (б)
a	int (*a)[][J][K]	96
a[0]	int (*a)[][J]	48
a[0][0]	int (*a)	16
a[0][0][0]	int *	4

Чтобы посмотреть дамп каждого из указателей, необходимо сначала узнать его размер. Так как массив – статический, то на хранение дополнительной информации не будет выделено байт, а значит сумма всех компонент `int * (в байтах)` должно совпадать с суммой всех компонент `int (*a)` и так далее. Покажем, что соседние компоненты занимают одинаковое количество байт:

(для ***a):

```
(gdb) print sizeof(***a)

$5 = 4

(gdb) x /4xb **a

0x7fffffffdec0: 0x01 0x00 0x00 0x00

(gdb) x /4xb **a+1

0x7fffffffdec4: 0x02 0x00 0x00 0x00

(gdb) x /4xb **a+2

0x7fffffffdec8: 0x03 0x00 0x00 0x00
```

Командой `print sizeof(***a)` мы узнали размер самого маленького компонента массива. Но почему мы далее обращаемся по указателю `**a`?

Так как `**a` - указатель на массив из К(4) элементов, то проходясь по адресам `**a, **a+1`, и т.д. мы обращаемся к элементам массива из К(4) элементов, а значит можем посмотреть их содержимое и адрес.

В дампе выше видно, что между адресами `**a` и `**a+1` как раз 4 байта (4 байт – размер типа `int`), а значения их как раз соответствуют значениям, которые мы задали в программе (`**a - 1, **a+1 - 2, **a - 3` и т.д.).

Аналогичным способом покажем для компонент `**a, *a, a`:

(для **a):

```
(gdb) print sizeof(**a)
$7 = 16
(gdb) x /16xb *a
0x7fffffffdec0: 0x01 0x00 0x00 0x00 0x02 0x00 0x00 0x00
0x7fffffffdec8: 0x03 0x00 0x00 0x00 0x04 0x00 0x00 0x00
(gdb) x /16xb *a+1
0x7fffffffded0: 0x05 0x00 0x00 0x00 0x06 0x00 0x00 0x00
0x7fffffffded8: 0x07 0x00 0x00 0x00 0x08 0x00 0x00 0x00
(gdb) x /16xb *a+2
0x7fffffffdee0: 0x09 0x00 0x00 0x00 0x0a 0x00 0x00 0x00
0x7fffffffdee8: 0x0b 0x00 0x00 0x00 0x0c 0x00 0x00 0x00
```

В данном дампе видно, что адреса соседних строк отличаются на 8, значит в каждой строке выведено 2 числа из массива, размерность которых - 4байт.

Последние числа каждой из компонент `*a, *a+1` и т.д. в десятичном представлении совпадают с реальными значениями из программы (`0x04 - 4; 0x08 - 8; 0x0c - 12`).

(для *a):

```
(gdb) print sizeof(*a)
$8 = 48
(gdb) x /48xb a
0x7fffffffdec0: 0x01 0x00 0x00 0x00 0x02 0x00 0x00 0x00
0x7fffffffdec8: 0x03 0x00 0x00 0x00 0x04 0x00 0x00 0x00
0x7fffffffded0: 0x05 0x00 0x00 0x00 0x06 0x00 0x00 0x00
0x7fffffffded8: 0x07 0x00 0x00 0x00 0x08 0x00 0x00 0x00
0x7fffffffdee0: 0x09 0x00 0x00 0x00 0x0a 0x00 0x00 0x00
0x7fffffffdee8: 0x0b 0x00 0x00 0x00 0x0c 0x00 0x00 0x00
(gdb) x /48xb a+1
0x7fffffffdef0: 0x0d 0x00 0x00 0x00 0x0e 0x00 0x00 0x00
0x7fffffffdef8: 0x0f 0x00 0x00 0x00 0x10 0x00 0x00 0x00
0x7fffffffdf00: 0x11 0x00 0x00 0x00 0x12 0x00 0x00 0x00
```

```
0x7fffffffdf08: 0x13  0x00  0x00  0x00  0x14  0x00  0x00  0x00
0x7fffffffdf10: 0x15  0x00  0x00  0x00  0x16  0x00  0x00  0x00
0x7fffffffdf18: 0x17  0x00  0x00  0x00  0x18  0x00  0x00  0x00
```

(для a):

```
(gdb) print sizeof(a)
$9 = 96
(gdb) x /96xb &a
0x7fffffffdec0: 0x01  0x00  0x00  0x00  0x02  0x00  0x00  0x00
0x7fffffffdec8: 0x03  0x00  0x00  0x00  0x04  0x00  0x00  0x00
0x7fffffffded0: 0x05  0x00  0x00  0x00  0x06  0x00  0x00  0x00
0x7fffffffded8: 0x07  0x00  0x00  0x00  0x08  0x00  0x00  0x00
0x7fffffffdee0: 0x09  0x00  0x00  0x00  0x0a  0x00  0x00  0x00
0x7fffffffdee8: 0x0b  0x00  0x00  0x00  0x0c  0x00  0x00  0x00
0x7fffffffdef0: 0x0d  0x00  0x00  0x00  0x0e  0x00  0x00  0x00
0x7fffffffdef8: 0x0f  0x00  0x00  0x00  0x10  0x00  0x00  0x00
0x7fffffffdf00: 0x11  0x00  0x00  0x00  0x12  0x00  0x00  0x00
0x7fffffffdf08: 0x13  0x00  0x00  0x00  0x14  0x00  0x00  0x00
0x7fffffffdf10: 0x15  0x00  0x00  0x00  0x16  0x00  0x00  0x00
0x7fffffffdf18: 0x17  0x00  0x00  0x00  0x18  0x00  0x00  0x00
```

Видно, что в дампах для (`a+1`) и (`&a`) совпадают концы (`0x18 = 0x18 = 24`). Таким образом, наш массив a, действительно, статический.

5.

```
#define I 2
#define J 3
#define K 4

// Число
void print1(int *a)
```

```
{
   printf("%ld\n", sizeof(*a));
}
// Массив размера K
void print2(int a[][K])
{
   printf("%ld\n", sizeof(*a));
}
// Массив из J массивов размера K
void print3(int a[][J][K])
{
   printf("%ld\n", sizeof(*a));
}
int main(void)
{
   int a[I][J][K] = {{{1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12}},
{{13, 14, 15, 16}, {17, 18, 19, 20}, {21, 22, 23, 24}}};
   print1(**a);
   print2(*a);
   print3(a);
      printf("%ld\n", sizeof(a));
   return OK;
}
```

Вывод программы:

```
4
16
48
96
```