

Технология программирования на ЭВМ

Динамические массивы

Баев А.Ж.

Казахстанский филиал МГУ

20 декабря 2018

Статическое выделение памяти

Как получить значения массива x вне функции f ?

```
#include <stdlib.h>

void f() {
    // allocate static memory for x
    int x[3];
    x[0] = 11;
    x[1] = 12;
    x[2] = 13;
    // free static memory for x
}

int main() {
    f();
    ???
}
```

Динамическое выделение памяти

Цель 1: выделить и удалить память в любой функции.

Цель 2: выделить столько места, сколько нужно.

```
#include <stdlib.h>

void *malloc(size_t n);
```

От английского «выделение памяти» (memory allocation).

Один байт

Выделить память

```
1 char *ptr;  
2 ptr = malloc(1);
```

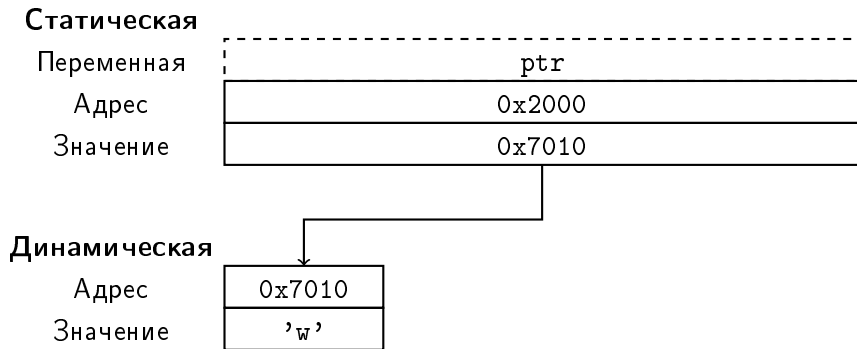
Записать значение

```
1 *ptr = 'w';
```

Очистить память

```
1 free(ptr);
```

Один байт



Динамическое выделение памяти

Как получить значения массива x вне функции f ?

```
#include <stdlib.h>
void f() {
    // allocate dynamic memory for x
    int *x;
    x = malloc(12);
    x[0] = 11;
    x[1] = 12;
    x[2] = 13;
}
int main() {
    int *a;
    a = f();
    free(a);
    // free dynamic memory for x
}
```

Динамическое выделение массивов

Сгенерировать динамический массив из 5 целых чисел типа `int`.

```
1  int *a = malloc(5 * sizeof(int));
```

Заполнить его цифрами от 10 до 14.

```
2  for (int i = 0; i < 5; i++)  
3      a[i] = 10 + i;
```

Распечатать массив.

```
4  for (int i = 0; i < 5; i++)  
5      printf("%d_", a[i]);
```

Очистить память.

```
6  free(a);
```

Динамическая память. Процедурный код

```
1  int *generate(int n) {
2      int *ptr = malloc(n * sizeof(int));
3      for (int i = 0; i < n; i++)
4          ptr[i] = i + 1;
5      return ptr;
6  }
7  void print(int n, int *ptr) {
8      for (int i = 0; i < n; i++)
9          printf("%d ", ptr[i]);
10 }
11 int main() {
12     int *ptr = generate(5);
13     print(5, ptr);
14     free(ptr);
15     return 0;
16 }
```


Динамическая память. Нужно больше функций

```
1 void *calloc(size_t n, size_t size);
```

Выделяет массив из n элементов, каждый из которых имеет размер $size$ байт (всего $n*size$ байт).

```
1 void *realloc(void *ptr, size_t size);
```

- ❶ изменяет размер массива $size$ (может выделить новую память);
- ❷ если новый размер больше старого, то добавляет ячейки с мусором, иначе отбрасывает лишние с конца;

Замечание: при $ptr = NULL$ работает как *malloc*.

Динамическая память. Скорость работы.

Работает быстрее чем вы думаете! Почему?

```
1  char *ptr = NULL;
2  for (size_t n = 1; n <= 1000000; n++) {
3      ptr = realloc(ptr, n);
4  }
```

Динамическая память vs статическая память.

1. именованные переменные.
2. время жизни.
3. место для памяти.
4. возможность узнать размера массива.
5. размер массива: $n * \text{sizeof}(\text{type})$ и $n * \text{sizeof}(\text{type}) + \text{sizeof}(\text{type}^*)$.
6. представление матриц.

Пример. Простые числа.

Дано целое число n от 1 до 100. В динамический массив записать все простые числа, которые не превосходят n . Вывести количество простых чисел и сами числа по возрастанию.

Ввод	8	2
Вывод	4 2 3 5 7	0

Пример. Простые числа.

```
1  int main() {
2      int n;
3      scanf("%d", &n);
4
5      int *ans = NULL, m = 0;
6      for (int i = 2; i <= n; i++)
7          if (isprime(i)) {
8              ans = realloc(ans, (m + 1) * sizeof(int));
9              ans[m] = i;
10             m++;
11         }
12
13     print(ans);
14
15     if (ans != NULL)
16         free(ans);
17     return 0;
18 }
```

Пример. Обратный порядок

Дана последовательность целых чисел от 1 до 10^9 . Ввод заканчивается нулём. Вывести их в обратном порядке.

Ввод	1 3 2 5 0
Вывод	5 2 3 1

Пример. Обратный порядок

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main() {
5      int *array = NULL, n = 0, value;
6
7      scanf("%d", &value);
8      while (value != 0) {
9          array = realloc(array, (n + 1) * sizeof(int));
10         array[n] = value;
11         n++;
12         scanf("%d", &value);
13     }
14
15     for (int i = n - 1; i >= 0; i--)
16         printf("%d_", array[i]);
17     putchar('\n');
18
19     if (array)
20         free(array);
21     return 0;
```

Пример. Фильтрация массива.

Дано целое положительное n . Далее n целых чисел. Вывести только четные числа.

Ввод	3 14 11 12
Вывод	14 12

Пример. Фильтрация массива.

Описать функции *scan*, *filter*, *print* без возвращаемого значения с таким интерфейсом:

```
1  int main() {
2      int *a, *b;
3      int n, m;
4      scan(&n, &a);
5
6      filter(n, a, &m, &b);
7      if (a != NULL)
8          free(a);
9
10     print(m, b);
11     if (b != NULL)
12         free(b);
13     return 0;
14 }
```

Пример. Считать массив.

```
1 void scan(int *n_ptr, int **a_ptr) {
2     int *array = NULL, size, i;
3     scanf("%d", &size);
4
5     array = malloc(size * sizeof(int));
6     for (i = 0; i < size; i++)
7         scanf("%d", &array[i]);
8
9     *n_ptr = size;
10    *a_ptr = array;
11 }
```

Пример. Отфильтровать массив.

```
1 void filter(int n, int *a,  
2             int *m_ptr, int **b_ptr) {  
3     int *b = NULL, m = 0;  
4  
5     for (int i = 0; i < m; i++)  
6         if (a[i] % 2 == 0) {  
7             b = realloc(b, (m + 1) * sizeof(int));  
8             b[m] = b[i];  
9             m++;  
10        }  
11  
12    *m_ptr = m;  
13    *b_ptr = b;  
14 }
```

Пример. Динамический scanf

Дана строка из слов. Слова разделены пробелами, каждое слово состоит из печатных символов, отличных от пробела, табуляции и переноса строки. Считать каждое слово в динамический массив. Вывести слова.

```
1 char *get_word(char *last_char_ptr);
```

Пример. Динамический scanf

```
1 char *get_word() {
2     char delimiter = ' ', final = '\n';
3     char *word = NULL, ch;
4     int n = 0;
5
6     ch = getchar();
7     while (ch != delimiter && ch != final) {
8         word = realloc(word, (n + 1) * sizeof(char));
9         word[n] = ch;
10        n++;
11        ch = getchar();
12    }
13
14    word = realloc(word, (n + 1) * sizeof(char));
15    word[n] = '\0';
16
17    return word;
18 }
```

Пример. Динамический scanf

```
1 int main() {  
2     char *word = get_word();  
3     while (word != NULL) {  
4         puts(word);  
5         free(word);  
6         word = get_word();  
7     }  
8     return 0;  
9 }
```

Динамическая память. Еще раз об ошибке сегментации

```
1  int *ptr = malloc(10);  
2  ptr[9] = 0;
```

Может и не произойти. Массивы выделяются со смещением по степеням двойки.

Динамическая память. Еще раз об ошибке сегментации

```
1  int *ptr = malloc(10);  
2  ptr[9] = 0;
```

Может и не произойти. Массивы выделяются со смещением по степеням двойки. Средство борьбы — санитайзеры.

Динамическая память. Совсем плохо

```
1   int a = 5, b = 7;  
2   int *ptr = &a;  
3   *(ptr + 1) = 6
```

```
1 gcc prog.c -o prog -fsanitize=address,undefined
```

Статическая матрица — разложенный по строкам массив

Разложенная по строкам в виде массива (плотная упаковка).

В программе

```
int a[2][3];  
a[0][0] = 1; a[0][1] = 2; a[0][2] = 3;  
a[1][0] = 4; a[1][1] = 5; a[1][2] = 6;
```

В памяти

Статическая

Переменная

Адрес

Значение

a					
2000	2004	2008	2012	2016	2020
1	2	3	4	5	6

Динамические матрицы — массив массивов

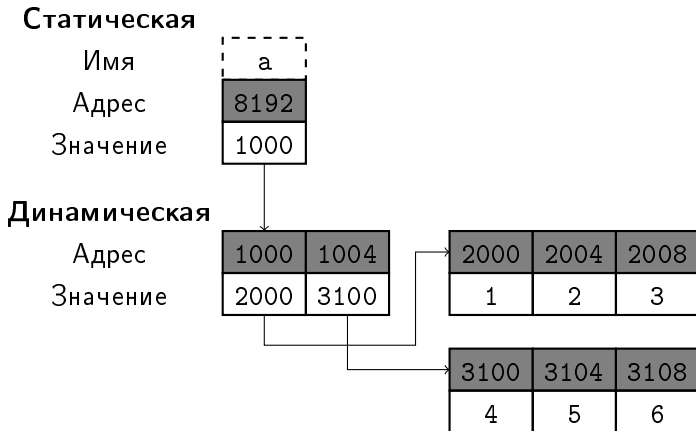
Массив указателей на динамические массивы (неплотная упаковка).

В программе

```
int **a = malloc(2 * sizeof(int *));  
a[0] = malloc(3 * sizeof(int));  
a[1] = malloc(3 * sizeof(int));  
a[0][0] = 1; a[0][1] = 2; a[0][2] = 3;  
a[1][0] = 4; a[1][1] = 5; a[1][2] = 6;
```

Динамические матрицы — массив массивов

В памяти



Динамические матрицы. Матрица 2x3

Очистка

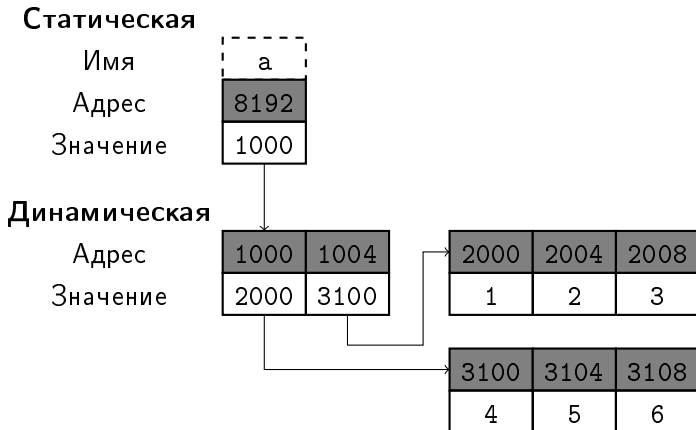
```
1    free(a[0]);  
2    free(a[1]);  
3    free(a);
```

Динамические матрицы. Любимая плюшка линала

Перестановка строк!

```
1  int *tmp = a[0];  
2  a[0] = a[1];  
3  a[1] = tmp;
```

Динамические матрицы. Матрица 2x3



Отличия динамической и статической матрицы.

	статическая	динамическая
память (размер $n \times m$)	$n \cdot m \cdot s$	$n \cdot m \cdot s + (1 + n) \cdot p$
упаковка	плотно вся матрица	плотно отдельные строки
перестановка строк	$O(n)$ действий	$O(1)$ действий
размеры строк	одинаковые	не обязательно одинаковые

Единичная матрица.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 int main() {
4     int n, **a;
5     scanf("%d", &n);
6
7     a = malloc(n * sizeof(int *));
8     for (int i = 0; i < n; i++)
9         a[i] = malloc(n * sizeof(int));
10
11     for (int i = 0; i < n; i++)
12         for (int j = 0; j < n; j++)
13             if (i == j)
14                 a[i][i] = 1;
15             else
16                 a[i][j] = 0;
```

Единичная матрица.

```
17     for (i = 0; i < n; i++) {
18         for (j = 0; j < n; j++)
19             printf("%2d", a[i][j]);
20         putchar('\n');
21     }
22
23     for (i = 0; i < n; i++)
24         free(a[i]);
25     free(a);
26     return 0;
27 }
```