# Лекция 10. Моделирование памяти Си-программы

# Цель лекции

Построить аксиоматику для памяти Си-программы.

# Содержание

# Зачем нужны указатели?

Правильно ли здесь используются указатели? Что делает этот код?

```
void print__intar(const int *v, size_t n)
{
    for (int i = 0; i < n; ++i) {
        printf("%d ", v[i]);
    }
}
```

# Зачем нужны указатели?

Правильно ли здесь используются указатели? Что делает этот код?

```
int vector__init(vector *v, size_t n)
{
    v->capacity = v->size = n;
    v->data = malloc(n * sizeof *v);
    return v->data ? 0 : 1;
}
```

# Зачем нужны указатели?

Правильно ли здесь используются указатели? Что делает этот код?

```
int *a1 = malloc(10 * sizeof *a1);
int *a2 = (int *)malloc(10 * sizeof *a2);
```

# Зачем нужны указатели?

Правильно ли здесь используются указатели? Что делает этот код?

```
char *strcpy(char *d, const char *s)
{
    char *r = d;
    while (*r++ = *s++) ;
    return d;
}
```

# Зачем нужны указатели?

Правильно ли здесь используются указатели? Что делает этот код?

```
char *memmove(void *d, const void *s, size_t n)
{
    if (d + n < s || s + n < d) {
        return memcpy(d, s, n);
    } else {
        void *t = malloc(n);
        memcpy(t, s, n);
        memcpy(d, t, n);
        free(t);
        return d;
    }
}
```

# Зачем нужны указатели?

Правильно ли здесь используются указатели? Что делает этот код?

```
typedef struct node {
    struct node *next;
    int data;
} Node;
void push(Node **list, int v)
{
    Node *head = malloc(sizeof *head);
    head->next = *list;
    head->data = v;
    *list = head;
}
```

# Зачем нужны указатели?

Правильно ли здесь используются указатели? Что делает этот код?

```
...
int fd[2];
pipe(fd);
...
struct { int a, b; } msg = {1, 2};
write(fd[1], &msg, sizeof msg);
...
```

# Содержание

# Примеры

Попробуйте придумать алгоритм построения модели программы, подходящий для всех следующих примеров.

# Пример 1

```
void f1(int *a, int n) { ... }
void f() {
    int mm[10];
    f1(&mm[3], 4);
}
```

Всегда ли параметр-указатель означает начало массива?

# Пример 2

```
void f2(int *a, int n, int *v) { ... }
```

Всегда ли параметр-указатель означает, что функция получает массив (отдельный блок памяти)?

# Пример 3

```
void f3(int *a1, int n1, int *a2, int n2) { ... }
```

Всегда ли эта функция получает два массива (отдельных блока памяти)?

# Пример 4

```
int Vector__resize(struct Vector *v, int newsize) {
    ... realloc ...
}
```

Как записать постусловие, что эта функция может аллоцировать новый блок памяти? (1 - что блок аллоцируется, 2 - что блока не было раньше) Какие входные данные нужны, чтобы записать это постусловие?

# Пример 5

```
int main(int argc, char *argv[]) { ... }
```

Сколько массивов (отдельных блоков памяти) нужно передать как входные данные этой функции?

# Выводы из примеров

- Кроме своих аргументов, функции получают неявно еще входные данные - блоки памяти.
- Количество блоков памяти иногда трудно, а иногда невозможно определить по тексту программы.

# Содержание

# Состав модели памяти

- массив – типизированный, поэтому блок – тоже типизированный
- поэтому для каждого типа нужна целиком отдельная модель памяти
- тип-символ memory - вся модель памяти
- тип-символ block - блок
- тип-символ pointer - указатель

# Соотношения

- каждый memory – множество из block-ов
- каждый block имеет массив array
- любой block может быть allocated или не allocated (allocable)
- любой pointer имеет block и индекс в его массиве
- любой pointer может быть равен NULL
- любые два pointer из разных block-ов не сравнимы

# Предикатные и функциональные символы, примитивы

- любой pointer является valid (валидным, корректно разыменуемым), если он указывает внутрь block, который allocated
- адресная арифметика: shift pointer, sub pointers
- примитивы для блоков: malloc, free
- примитивы для указателей (разыменование): write, read

# Первые определения - первые проблемы

- Попробуйте определить спецификацию для read - должна нормально получиться
- Попробуйте определить спецификацию для write - получится слишком много кванторов! Как их уменьшить? Переделать модель памяти.
- Надо иметь всего один массив (для значений по указателям). Как тогда определить индексы и адреса начала блока? Массив надо заменить просто на отображение указателей в значения.
- И отделить множество блоков от этого отображения. Получаются 2 отдельных типа, совместно представляющих модель памяти: отображение и множество блоков с их статусом.

# Определяем отображения

```
theory Map
  type map 'a 'b
  function get (m: map 'a 'b) (v: 'a): 'b
  function set (m: map 'a 'b) (v: 'a) (v2: 'b)
  axiom get_set: forall m: map 'a 'b,
      a a2: 'a, b: 'b.
      get (set m a b) a2 = (if a = a2 then b else
        get m a2)
end
```

# Типы

```
type block
type pointer 't = (block, int)
type alloc_table 't = Map.map block int
(* alloc_table[block] >= 0 -- block is allocated
   and here is its size;
   alloc_table[block] < 0 -- block is allocable *)
type memory 't = Map.map (pointer 't) 't
```

# Дополнительные символы

```
function blockOf (p: pointer 't): block
  = let (b, o) = p in b
function offsetOf (p: pointer 't): int
  = let (b, o) = p in o
function getSz (a: alloc_table 't) (b: block): int
  = Map.get a b
function setSz (a: alloc_table 't) (b: block)
 (sz: int): alloc_table 't = Map.set a b sz
function getV (m: memory 't) (p: pointer 't): 't
  = Map.get m p
function setV (m: memory 't) (p: pointer 't)
 (v: 't): memory 't = Map.set m p v
```

# Дополнительные символы

```
function blockLen (a: alloc_table 't)
    (p: pointer 't): int = getSz a (blockOf p)
predicate allocable (a: alloc_table 't)
                    (p: pointer 't)
    = blockLen a p < 0
predicate allocated (a: alloc_table 't)
                    (p: pointer 't)
    = not (allocable a p)
predicate freeable (a: alloc_table 't)
                   (p: pointer 't)
    = allocated a p /\ offsetOf p = 0
```

# Дополнительные символы

```
predicate valid (a: alloc_table 't) (p: pointer 't)
    =  0 <= offsetOf p < blockLen a p
predicate valid_range (a: alloc_table 't)
                      (p: pointer 't) (n: int) =
     0 <= offsetOf p < blockLen a p - n
predicate same_block (p p2: pointer 't) =
    blockOf p = blockOf p2
function shiftP (p: pointer 't)(n: int): pointer 't
 = let (b, o) = p in (b, o + n)
function subP (p p2: pointer 't): int
 = let (b, o) = p in let (b2, o2) = p2 in o - o2
```

# Примитивы для операторов CALL

```
val readPtr (a: alloc_table 't) (m: memory 't)
            (p: pointer 't): 't
    requires { valid a p }
    ensures { result = getV m p }
val writePtr (a: alloc_table 't) (m: memory 't)
             (p: pointer 't) (v: 't): memory 't
    requires { valid a p }
    ensures { result = setV m p v }
val shiftPtr (p: pointer 't) (n: int): pointer 't
    ensures { same_block p result }
    ensures { offsetOf result = offsetOf p + n }
val subPtr (p: pointer 't) (p2: pointer 't): int
    requires { same_block p p2 }
    ensures { result = offsetOf p - offsetOf p2 }
```

# Примитивы (2)

```
val malloc (a: alloc_table 't) (m: memory 't)
    (sz: int): (alloc_table 't, pointer 't)
    requires { sz >= 0 }
    ensures { let (a2, p) = result in
        blockOf p was allocable
        p is freeable
        blockLen p = sz
        sizes of other blocks are not changed
        allocation statuses of other blocks
              are not changed
    }
```

# Примитивы (3)

```
val free (a: alloc_table 't) (m: memory 't)
    (p: pointer 't): alloc_table 't
    requires { p is freeable }
    ensures {
        blockOf p is allocable
        sizes of other blocks are not changed
        allocation statuses of other blocks
              are not changed
    }
```

# Пример

Проверьте эффективность этой аксиоматизации модели памяти. Для этого решите следующие задания.

- Напишите на языке Си задачу прошлой лекции с массивом и заменой там одного значения на другое. Напишите модель программы. Напишите спецификацию. Докажите полную корректность модели программы относительно этой спецификации.

# Вопросы

- Как промоделировать `null`?
- Как промоделировать Си-структуры? Массивы из Си-структур?
- Как промоделировать неявное приведение `void *` к любому указателю?