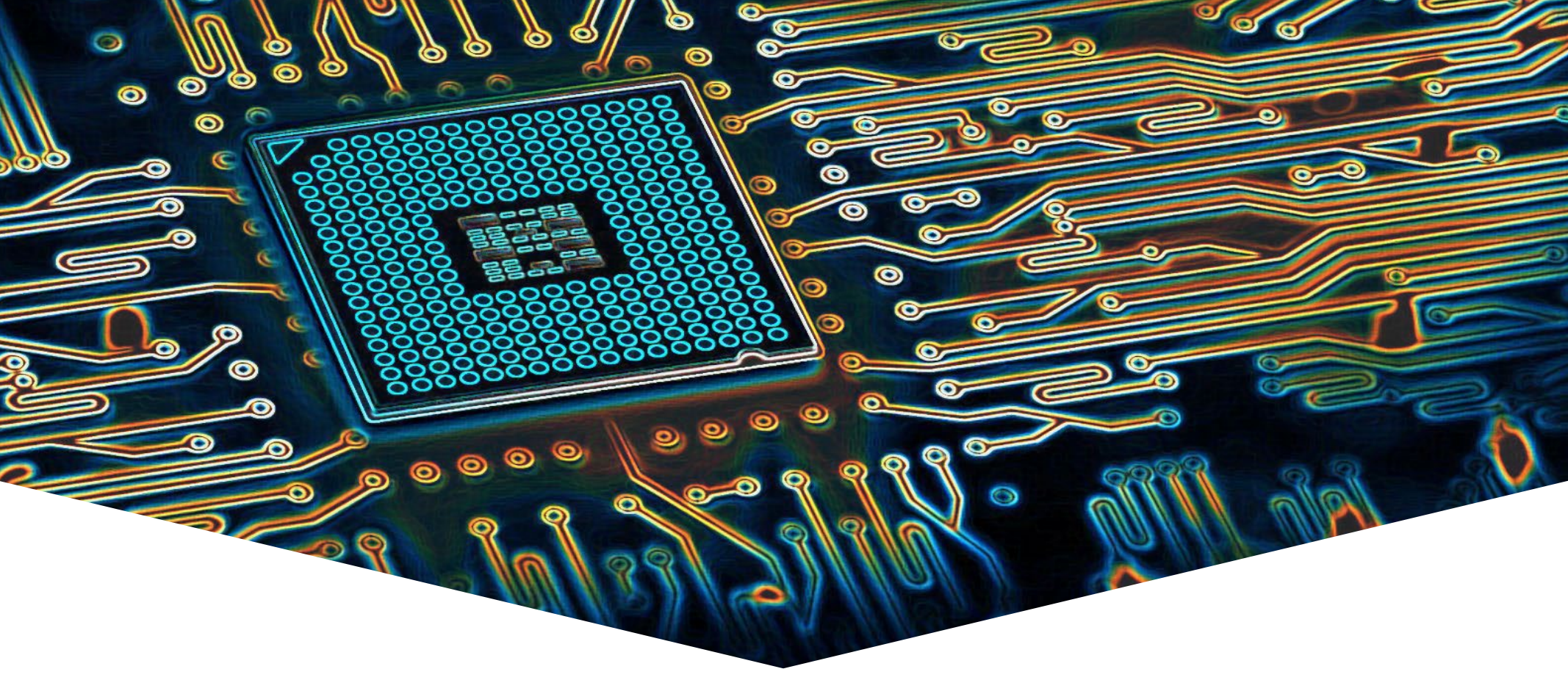


EMS31 Kwartaal 3 Week 3: Dynamisch geheugenallocatie

Dynamisch geheugenallocatie

Leerdoelen week 1 les 2. Je leert hoe je:

- dynamisch geheugen kunt alloceren en weer vrij kunt geven;
- een FIFO-buffer kunt implementeren die dynamisch groeit en krimpt met behulp van een singly linked list;
- een FIFO-buffer als een user defined type kunt definiëren waardoor je meerdere buffers in een programma kunt gebruiken.

Dynamische datastructuren

Probleem:

- Array en `struct` zijn statische datastructuren, ze hebben een vaste grootte en structuur.

Doel:

- Een dynamische datastructuur maken die run-time kan groeien en krimpen.

Heeft de taal C geen dynamische datastructuren?

Does standard c library provides linked list etc. data structures?

Asked 7 years, 4 months ago Active 2 years ago Viewed 58k times

50

Do standard C library implementations, especially **glibc** (the GNU C Library) provide linked lists, stack et al. data structures, or do we have to roll our own?

Thanks.

c gcc data-structures glibc

14

share edit follow flag

edited Aug 7 '16 at 23:02
James Ko
20.8k ● 17 ● 72 ● 167

asked Dec 22 '12 at 9:27
rsjethani
1,853 ● 4 ● 22 ● 28

6 Answers

Active | Oldest | Votes

22

The C Standard does not provide data structures like linked list and stack.Some compiler implementations might provide their own versions but their usage will be non portable across different compilers.

So Yes, You have to write your own.

Bron: https://stackoverflow.com/questions/14001652

Hoe bouw je dynamische datastructuren in C?

Ingrediënten:

- structs;
  - Zie EMS20_week2_lab1.pptx
  - Zie Dictaat-C
- pointers;
  - Zie EMS20_week1_lab2.pptx
- standaard functies `malloc` en `free`.

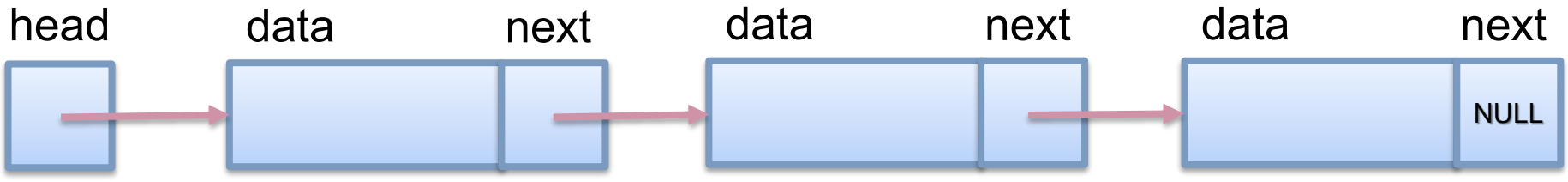
Welke datastructuren zijn er allemaal?

V · T · E	Data structures [hide]
Types	Collection · Container
Abstract	Associative array (Multimap) · List · Stack · Queue (Double-ended queue) · Priority queue (Double-ended priority queue) · Set (Multiset · Disjoint-set)
Arrays	Bit array · Circular buffer · Dynamic array · Hash table · Hashed array tree · Sparse matrix
Linked	Association list · Linked list · Skip list · Unrolled linked list · XOR linked list
Trees	B-tree · Binary search tree (AA tree · AVL tree · Red–black tree · Self-balancing tree · Splay tree) · Heap (Binary heap · Binomial heap · Fibonacci heap) · R-tree (R* tree · R+ tree · Hilbert R-tree) · Trie (Hash tree)
Graphs	Binary decision diagram · Directed acyclic graph · Directed acyclic word graph
List of data structures	

Bron: https://en.wikipedia.org/wiki/List_of_data_structures

Voorbeelden van dynamische datastructuren

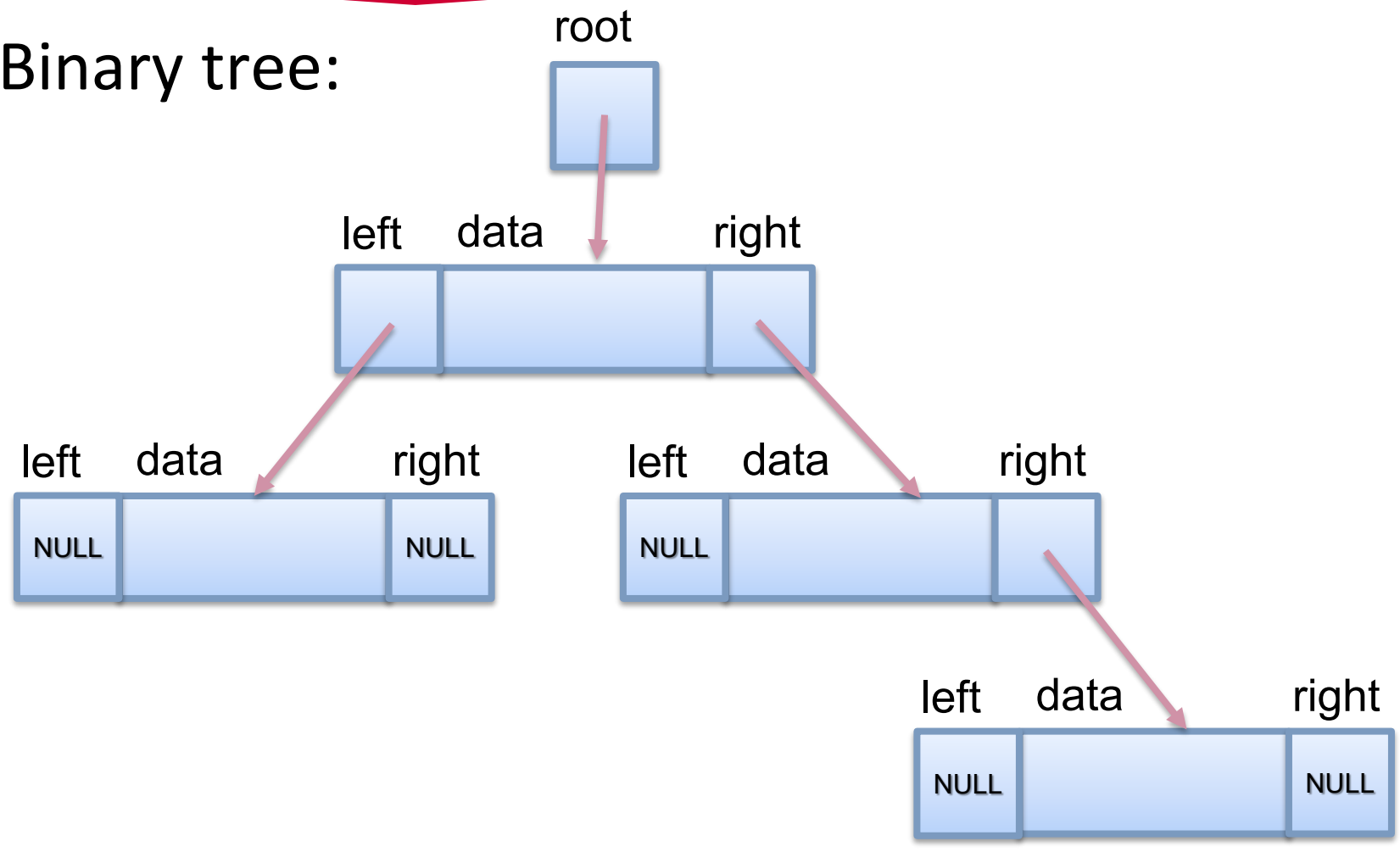
Singly linked list:



Zie voor animaties: Visual Algorithms

Voorbeelden van dynamische datastructuren

Binary tree:



Nodes

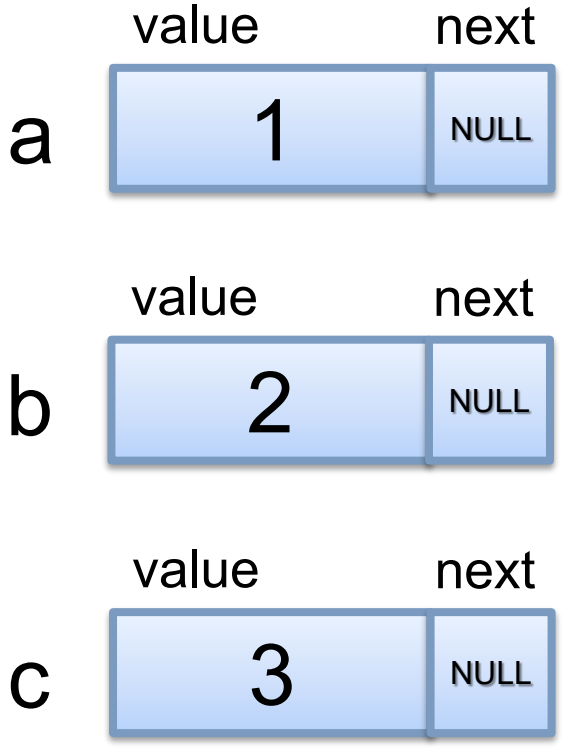
Een node is een `stuct` met:

- een of meer zelf gekozen datavelden;
- een of meer pointers (de links) naar een `struct` van hetzelfde type.

```
typedef struct ListNodeTag {
    int value;
    struct ListNodeTag *next;
} ListNode;
```

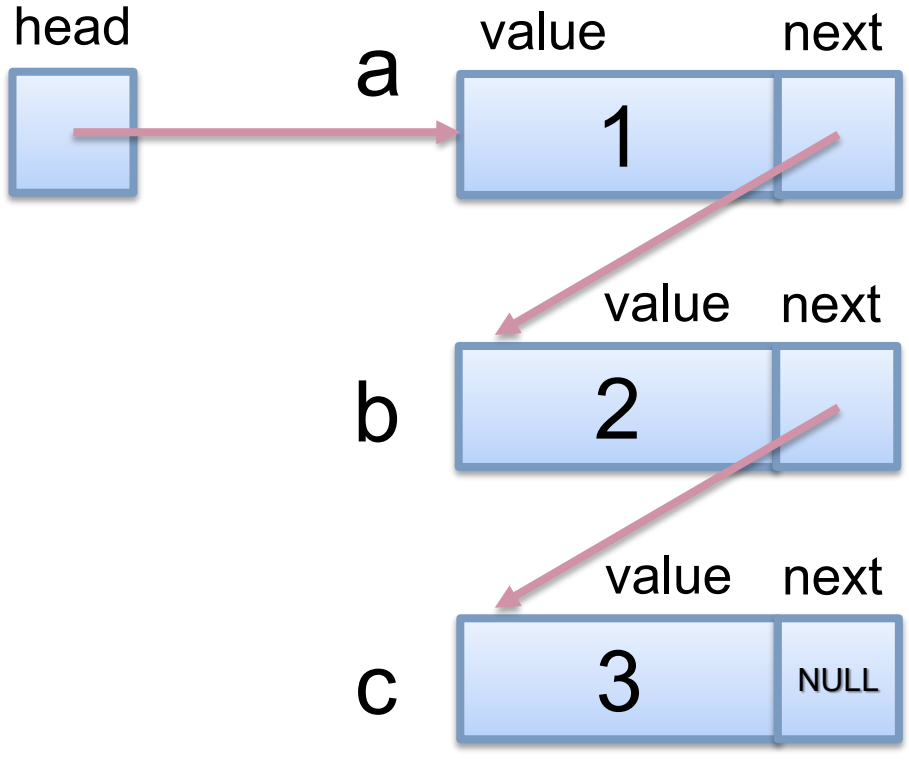
Voorbeeld singly linked list


```
typedef struct ListNodeTag {
    int value;
    struct ListNodeTag *next;
} ListNode;

//...

int main(void)
{
    ListNode a, b, c;
    a.next = b.next = c.next = NULL;
    a.value = 1; b.value = 2; c.value = 3;
```



Voorbeeld singly linked list


```
ListNode* head = &a;
a.next = &b;
b.next = &c;

printList(head);
```



Voorbeeld singly linked list


```
void printList(ListNode* n)
{
    while (n != NULL)
    {
        printf("%d", n->value);
        if (n->next != NULL)
        {
            printf(" --> ");
        }
        n = n->next;
    }
    printf("\n");
}
```

Output:

```
1 --> 2 --> 3
```

Deze singly linked list is nog steeds **statisch**. Hoe maken we hem dynamisch?

Zie: linked_list_static.c

Dynamische geheugenallocatie

Gebruik `malloc` om geheugen aan te vragen.

- Geheugen wordt door OS gereserveerd op de heap (deel van het RAM gereserveerd voor het betreffende proces).
- `malloc` geeft een `void *` terug die je zelf kunt toekennen aan het juiste pointer type.
- Extra informatie: Link

```
#include <stdlib.h>
//...

int main(void)
{
    ListNode *newNode = malloc(sizeof(ListNode));
```

Voorbeeld dynamic singly linked list


```
int main(void) {
    ListNode *head = NULL, *tail = NULL;
    int aantalNodes;

    printf("Hoeveel nodes wenst u? ");
    scanf("%d", &aantalNodes);

    for (int n = 0; n < aantalNodes; n++) {
        ListNode *newNode = malloc(sizeof(ListNode));
        newNode->value = n;
        newNode->next = NULL;
        if (head == NULL) {
            head = tail = newNode;
        }
        else {
            tail = tail->next = newNode;
        }
    }
    printList(head);
```

Output:

```
Hoeveel nodes wenst u? 5
0 --> 1 --> 2 --> 3 --> 4
```

Zie: linked_list_dynamic.c

Voorkom memory leaks!

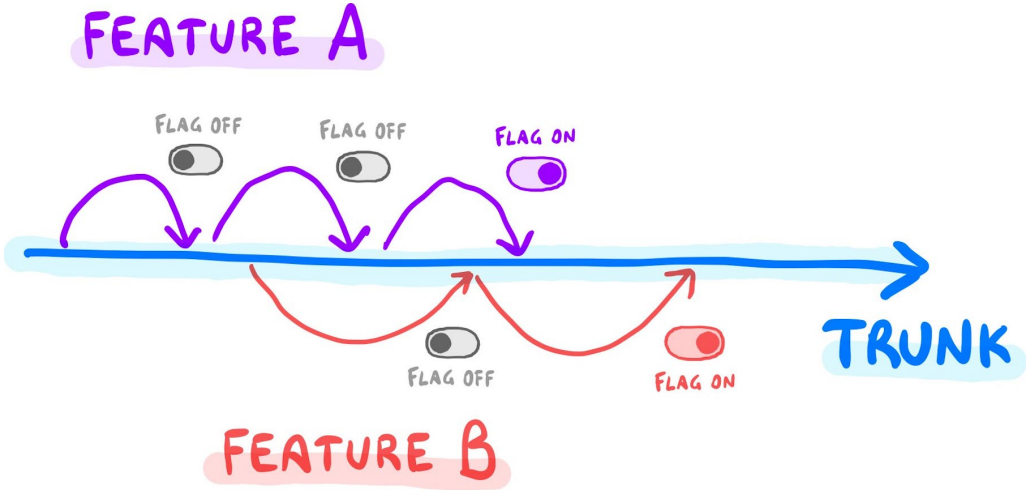
Gebruik `free` om geheugen weer vrij te geven (gebeurt ook automatisch als proces eindigt).

```
while (head != NULL)
{
    ListNode *node = head;
    head = head -> next;
    free(node);
}
tail = NULL;
```

"Hey! Your application has a memory leak."

Volgende les…

Git Trunk-based development met Feature flags



Aan de slag!

Aan de slag met [Opdrachten_Week_1.3.pdf]