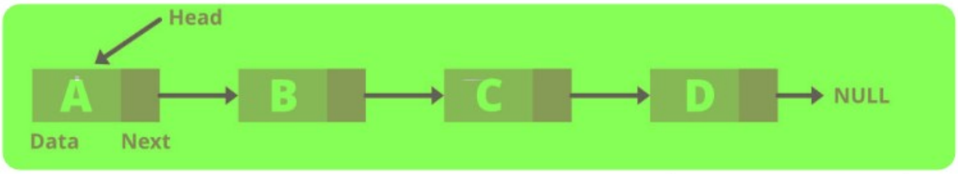
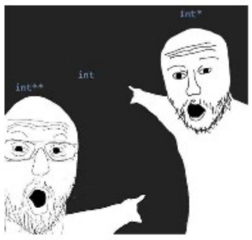
LISTA - SIMPLE

venerdì 23 dicembre 2022 16:53

SIMPLE LINKED LIST



```
struct node_t {
  int data;
  node_t * next;
};
```



Posso implementare qualsiasi algoritmo in C/C++ per implementarla!

Se voglio eliminare C, collego B a D!: Ponte!

Lista di stringhe

Io non conosco la dimensione della mia stringa e non ho possibilità di definire il nodo come ho fatto prima!

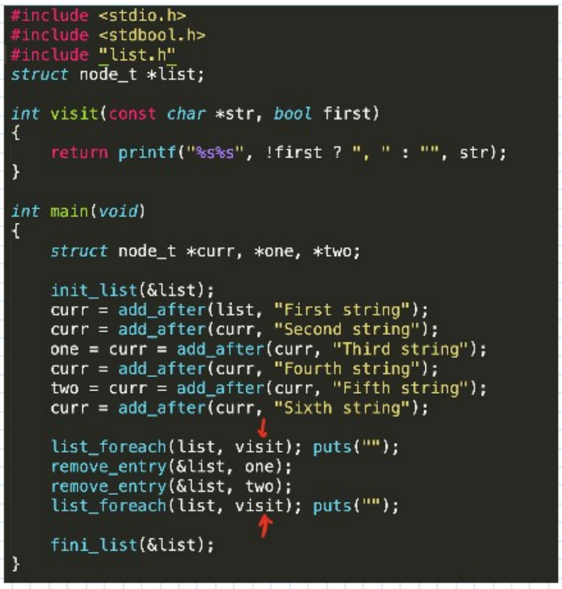
Come faccio a realizzare un nodo di lista a dimensione variabile? Vettori flessibili

E poi un puntatore a next.

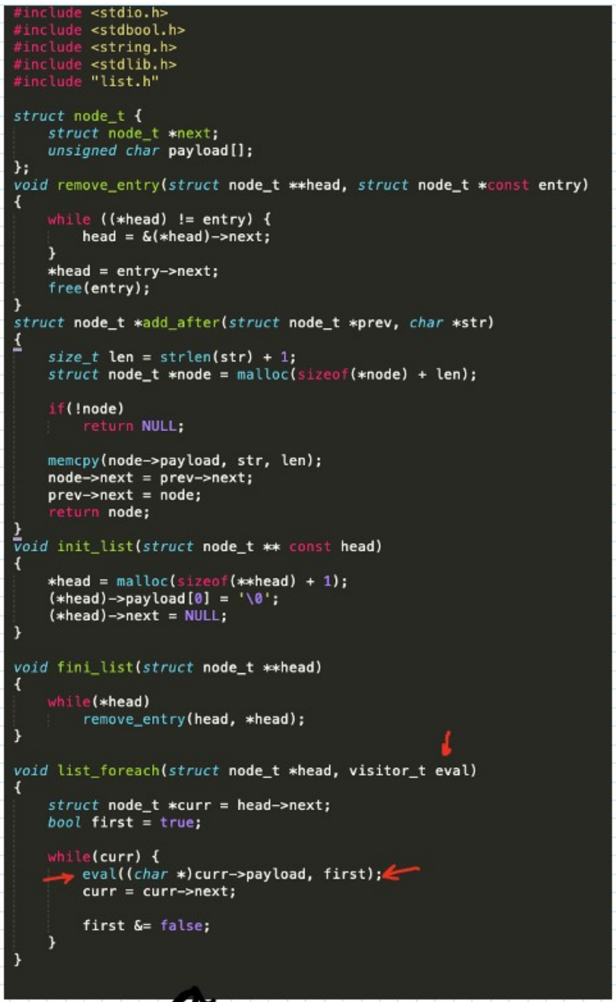
```
struct node_t {
        struct node_t *next;
        unsigned char payload[];
};

struct node_t *add_after(struct node_t *prev, char *str)
{
        size_t len = strlen(str) + 1;
        struct node_t *node = malloc(sizeof(*node) + len);
        memcpy(node->payload, str, len);
        node->next = prev->next;
        prev->next = node;

        return node;
}
```

File STRING.

MAIN.C   LIST.C

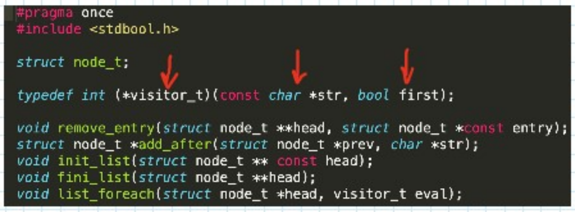


```c
#include <stdio.h>
#include <stdbool.h>
#include "list.h"
struct node_t *list;

int visit(const char *str, bool first)
{
    return printf("%s%s", !first ? ", " : "", str);
}

int main(void)
{
    struct node_t *curr, *one, *two;

    init_list(&list);
    curr = add_after(list, "First string");
    curr = add_after(curr, "Second string");
    one = curr = add_after(curr, "Third string");
    curr = add_after(curr, "Fourth string");
    two = curr = add_after(curr, "Fifth string");
    curr = add_after(curr, "Sixth string");

    list_foreach(list, visit); puts("");
    remove_entry(&list, one);
    remove_entry(&list, two);
    list_foreach(list, visit); puts("");

    fini_list(&list);
}
```



```c
#include <stdio.h>
#include <stdbool.h>
#include <string.h>
#include <stdlib.h>
#include "list.h"

struct node_t {
    struct node_t *next;
    unsigned char payload[];
};
void remove_entry(struct node_t **head, struct node_t *const entry)
{
    while ((*head) != entry) {
        head = &(*head)->next;
    }
    *head = entry->next;
    free(entry);
}
struct node_t *add_after(struct node_t *prev, char *str)
{
    size_t len = strlen(str) + 1;
    struct node_t *node = malloc(sizeof(*node) + len);

    if(!node)
        return NULL;

    memcpy(node->payload, str, len);
    node->next = prev->next;
    prev->next = node;
    return node;
}
void init_list(struct node_t ** const head)
{
    *head = malloc(sizeof(**head) + 1);
    (*head)->payload[0] = '\0';
    (*head)->next = NULL;
}

void fini_list(struct node_t **head)
{
    while(*head)
        remove_entry(head, *head);
}

void list_foreach(struct node_t *head, visitor_t eval)
{
    struct node_t *curr = head->next;
    bool first = true;

    while(curr) {
        eval((char *)curr->payload, first);
        curr = curr->next;

        first &= false;
    }
}
```

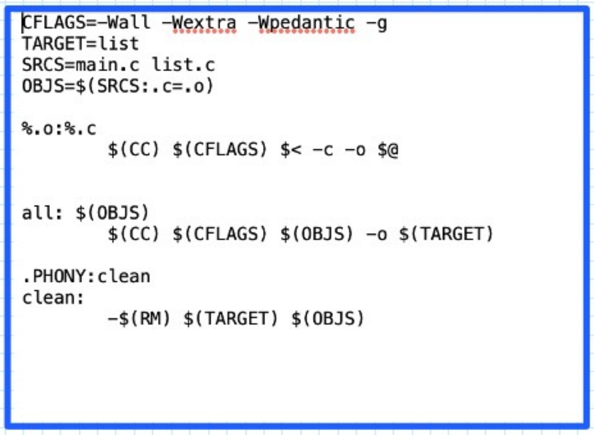
LIST. H



```c
#pragma once
#include <stdbool.h>

struct node_t;

typedef int (*visitor_t)(const char *str, bool first);

void remove_entry(struct node_t **head, struct node_t *const entry);
struct node_t *add_after(struct node_t *prev, char *str);
void init_list(struct node_t ** const head);
void fini_list(struct node_t **head);
void list_foreach(struct node_t *head, visitor_t eval);
```

dati astratti, ci scrivo solo
le firme, per modificare
le implementazioni modifico questo file

MAKEFILE:



```make
CFLAGS=-Wall -Wextra -Wpedantic -g
TARGET=list
SRCS=main.c list.c
OBJS=$(SRCS:.c=.o)

%.o:%.c
	$(CC) $(CFLAGS) $< -c -o $@


all: $(OBJS)
	$(CC) $(CFLAGS) $(OBJS) -o $(TARGET)

.PHONY:clean
clean:
	-$(RM) $(TARGET) $(OBJS)
```

Come faccio ad evitare che questo HEADER venga incluso più volte dal mio programma?

#PRAGMA ONCE

dico al preprocessore di includere solo una volta l'header in altri file!
IL file .h verrà inserito al più una volta in ciascun file C!