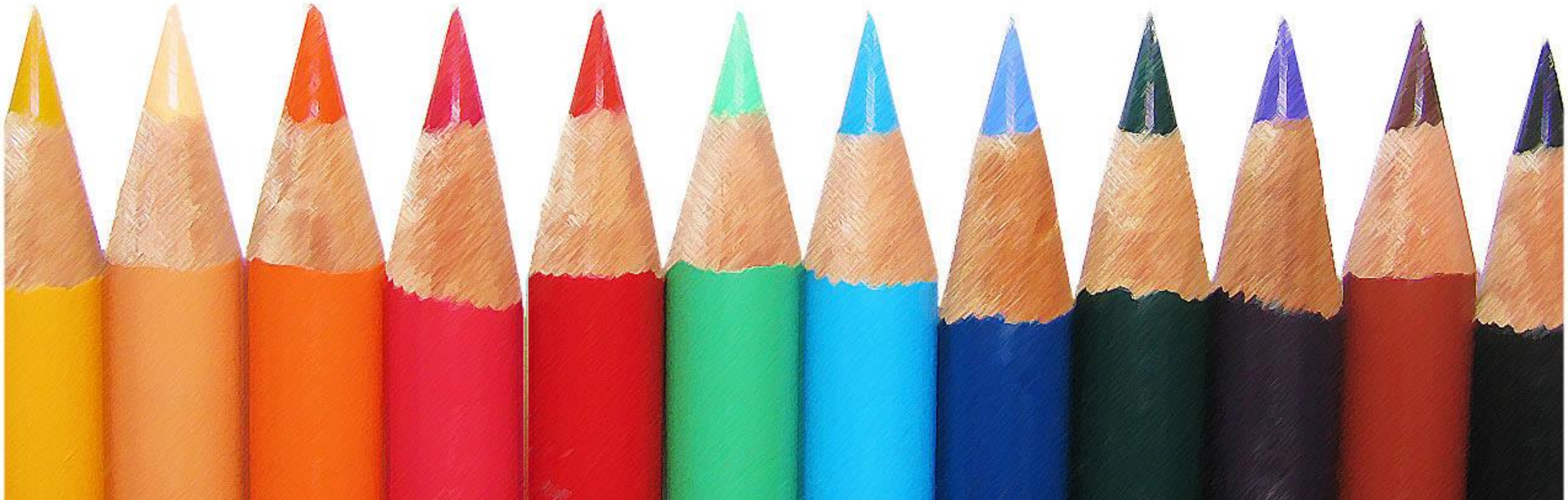
# *Dinamička alokacija memorije*



- U većini realnih aplikacija u trenutku pisanja programa nije moguće precizno predvideti memorijske zahteve programa, jer oni zavise od interakcije sa korisnikom i tek u fazi izvršavanja programa korisnik svojim akcijama implicitno određuje potrebne memorijske zahteve.

- Rešenje za to je dinamička alokacija memorije koja omogućava da program u toku rada zahteva određenu količinu memorije.
- Standardna biblioteka jezika C podržava dinamičko upravljanje memorijom kroz nekoliko funkcija koje su definisane u zaglavlju <stdlib.h>

# *Funkcija malloc*

*void *malloc(size_t n)*

- malloc alocira blok memorije veličine n bajtova i vraća adresu alociranog bloka memorije u vidu pokazivača,tipa void*.
- U slučaju da zahtev za memorijom nije moguće ispuniti ova funkcija vraća NULL.
- Memorija koju funkcija malloc vrati nije inicijalizovana i njen sadržaj je slučajan tj zavisi od podataka koji su ranije bili čuvani u tom delu memorije.

# *Funkcija calloc*

*void *calloc(size_t n, size_t size)*

- calloc vraća pokazivač na blok memorije veličine n objekata navedene veličine size.
- U slučaju da zahtev nije moguće ispuniti vraća se NULL.
- Za razliku od malloc-a memorija je inicijalizovana na nulu.

- Nakon poziva funkcija malloc() i calloc() poželjno je proveriti povratnu vrednost kako bi se proverilo da li je alokacija uspela.
- Kada nam dinamički alociran blok memorije više nije potreban, poželno je osloboditi ga. To se postiže funkcijom:

  *void free(void *p)*

- Poziv free(p) oslobađa memoriju na koju pokazuje pokazivač p, pri čemu je neophodno da p pokazuje na blok memorije koji je alociran pozivom funkcije malloc ili calloc.

- Velika je greška ako pokušamo osloboditi memoriju koja nije alocirana na ovaj način. Takodje ne sme se koristiti nešto što je već oslobođeno niti se sme dva puta oslobađati ista memorija.
- Redosled oslobađana memorije ne mora odgovarati redosledu alociranja.

# *Funkcija realloc*

- U nekim slučajevima potrebno je promeniti veličinu već alociranog bloka memorije. To se postiže funkcijom realloc:

  *void *realloc(void *memblock, size_t size)*
- Parametar memblock je pokazivač na prethodno alociran blok memorije, a parametar size je nova veličina u bajtovima.
- Funkcija realloc vraća pokazivač tipa void* na realociran blok memorije ili NULL u slučaju da zahtev ne može biti ispunjen.

# *Primer za malloc:*

```
#include <stdio.h>
#include <stdlib.h>
int main() {
int n, i, *a;

/* Unos broja elemenata*/
scanf(“%d”, &n);
/* Alocira se memorija */
if (( a = (int*)malloc(n*sizeof(int))) == NULL) {
    printf(“Greska prilikom alokacije memorije\n”);
    return 1;
}
```

```
/* Unos elemenata */
for ( i = 0; i<n; i++)
/* Ispis elemenata u obrnutom poretku */
for( i = n-1; i >= 0 ; i-- )
    printf("%d", a[i]);
/* Oslobadjanje memorije */
free(a);

return 0;
}
```

# *Primer za calloc:*

```
#include <stdio.h>
#include <stdlib.h>
#define br_elem 10
int main() {
int *m, *c, i;

m = malloc(br_elem*sizeof(int));
c = calloc(br_elem, sizeof(int));
for(i=0; i< br_elem; i++)
   printf("m[%d]=%d\n", i, m[i]);
for(i=0;i<br_elem;i++)
    printf("c[%d]=%d\n", i, c[i]);
free(m); free(c); }
```

# *Primer za realloc:*

```
#include <stdio.h>
#include <stdlib.h>
#define KORAK 10
int main() {
int *a = NULL;
int duzina = 0, alocirano = 0;
int n, i;
do {
   printf("Unesi ceo broj (-1 za kraj): ");
   scanf("%d", &n);
If ( duzina == alocirano ) {
alocirano = alocirano + KORAK;
a = realloc(a, alocirano*sizeof(int)); }
a[duzina++] = n;
}
```

```
while( n != -1);
printf(“ Uneto je %d brojeva. Alocirano je ukupno %d bajtova\n”, duzina, alocirano*sizeof(int));
printf(“ Brojevi su: “);
for(i=0;i<duzina;i++)
    printf(“%d”,a[i]);
free(a);
return 0;
}
```

# *Zadaci:*

1. Napisati program u C-u koji pronalazi sumu n elemenata koje unosi korisnik.

```
#include <stdio.h>
#include <stdlib.h>
 int main() {
 int n, i, *m, sum = 0;
 printf(„Unesite broje elemenata: ");
 scanf("%d",&n);
 m=(int*)malloc(n*sizeof(int)); //memorija se alocira
pomocu malloc-a if(m == NULL)  {
    printf(„Greska! Memorija nije alocirana.");
    exit(0);
}
```

```
printf(„Unesite elemente niza: ");
for(i = 0; i<n ; ++i)  {
    scanf("%d",m+i);
    sum += *(m+i);
 }
 printf("Suma = %d", sum);
 free(m);
 return 0;
}
```