calloc() versus malloc() in C

calloc()

The function calloc() stands for contiguous location. It works similar to the malloc() but it allocate the multiple blocks of memory each of same size.

Here is the syntax of calloc() in C language,

```
void *calloc(size_t number, size_t size);
```

Here,

number − The number of elements of array to be allocated.

size − Size of allocated memory in bytes.

Here is an example of calloc() in C language,

Example

Live Demo

```
#include <stdio.h>
#include <stdlib.h>
int main() {
   int n = 4, i, *p, s = 0;
   p = (int*) calloc(n, sizeof(int));
   if(p == NULL) {
      printf("\nError! memory not allocated.");
      exit(0);
   }
   printf("\nEnter elements of array : ");
   for(i = 0; i < n; ++i) {
      scanf("%d", p + i);
      s += *(p + i);
   }
   printf("\nSum : %d", s);
   return 0;
}
```

Output

```
Enter elements of array : 2   24    35   12
Sum : 73
```

In the above program, The memory block is allocated by calloc(). If the pointer variable is null, there is no memory allocation. If pointer variable is not null, user has to enter four elements of array and then sum of elements is calculated.

```
p = (int*) calloc(n, sizeof(int));
if(p == NULL) {
   printf("\nError! memory not allocated.");
```

```
   exit(0);
}
printf("\nEnter elements of array : ");
for(i = 0; i < n; ++i) {
   scanf("%d", p + i);
   s += *(p + i);
}
```

malloc()

The function malloc() is used to allocate the requested size of bytes and it returns a pointer to the first byte of allocated memory. It returns null pointer, if fails.

Here is the syntax of malloc() in C language,

```
pointer_name = (cast-type*) malloc(size);
```

Here,

pointer_name − Any name given to the pointer.

cast-type − The datatype in which you want to cast the allocated memory by malloc().

size − Size of allocated memory in bytes.

Here is an example of malloc() in C language,

Example

Live Demo

```
#include <stdio.h>
#include <stdlib.h>
int main() {
   int n = 4, i, *p, s = 0;
   p = (int*) malloc(n * sizeof(int));
   if(p == NULL) {
      printf("\nError! memory not allocated.");
      exit(0);
   }
   printf("\nEnter elements of array : ");
   for(i = 0; i < n; ++i) {
      scanf("%d", p + i);
      s += *(p + i);
   }
   printf("\nSum : %d", s);
   return 0;
}
```

Here is the output,

Output

```
Enter elements of array : 32 23 21 8
Sum : 84
```

In the above program, four variables are declared and one of them is a pointer variable *p which is storing the memory allocated by malloc. Elements of array are printed by user and sum of elements is printed.

```
int n = 4, i, *p, s = 0;
p = (int*) malloc(n * sizeof(int));
if(p == NULL) {
   printf("\nError! memory not allocated.");
   exit(0);
}
printf("\nEnter elements of array : ");
for(i = 0; i < n; ++i) {
   scanf("%d", p + i);
   s += *(p + i);
}
printf("\nSum : %d", s);
```