

Динамично резервиране на памет

Динамичен клас памет – сравнение с автоматичен и статичен `stdlib.h`

void * malloc(size_t sizeBt); случайни стойности;

/ масив от цели числа задаван от arr с размер cnt елемента
типът на аргумента size_t се дефинира в stdlib.h и е unsigned type */*

```
int cnt, *arr=NULL; // cnt – определя се в програмата
printf("Enter array length");
scanf("%d",&cnt); // контрол на входа !!
arr= (int*)malloc(cnt*sizeof(int)); //трябва да се преобразува до указател от съответния тип
// и да се изчислят необходимия брой байтове
if(arr==NULL){ printf("Memory allocation error"); exit(2); } //неуспешно
arr[0]=5; *(arr+1)=7;
```

void * calloc(size_t nel, size_t sizeEl); нулирани стойности

/ масив от цели числа задаван от arr с размер cnt елемента*/*

```
int cnt, *arr=NULL;
printf("Enter array length");
scanf("%d",&cnt); // контрол на входа !!
arr= (int*)calloc(cnt,sizeof(int));
if(arr==NULL){ printf("Memory allocation error"); exit(3); }
```

void free(void *); освобождава запазената памет, непредвидим ако паметта не е запазена

```
int* arr=NULL;
arr = (int*)malloc(10*sizeof(int));
if(arr) //контрол дали е успешно
free(arr);
```

void * realloc(void *, size_t sizeBt); Ако е успешно, връща указател към памет, която съдържа същите данни като старата област с новия размер. Ако новият размер е по-малък, останалата се освобождава, ако е по-голям при нужда се копира и се освобождава старата памет. Ако е неуспешно връща null стойност и не освобождава старата.

```
int cnt,cnt2
int* arr = NULL, *tmp;
do {
printf("Enter a positive integer: ");
scanf("%d", &cnt);
}while(cnt < 1);
arr = (int*)calloc(cnt, sizeof(int));
if(arr==NULL){ printf("Memory allocation error"); exit(2);}

/* масив от цели числа задаван от arr с променен размер cnt2 елемента*/
do {
printf("Enter another positive integer: ");
scanf("%d", &cnt2);
}while(cnt2 < 1);
tmp = (int*)realloc(arr, cnt2*sizeof(int));
if (tmp ==NULL) { free(arr); printf("Memory allocation error"); exit(2);}
arr=tmp;
```

Пример с динамичен масив от цели числа. Числата са записани във файл, четат се до края на файла и се въвеждат в динамичен масив от цели числа. Всеки път при липса на място в масива той увеличава размера си с DELTA на брой елемента. Програмата може да бъде свалена от [ТУК](#).

```
#include <stdio.h>
#include <stdlib.h>
#define DELTA 10 /* брой елементи, с които се увеличава масива при липса на място */
int* include(int digit,int* p,int *sizeAd,int* nm)
{
    int *tmp;
    if(!p){
        *sizeAd=DELTA;
        if((p=(int *)malloc((*sizeAd)*sizeof(int)))==NULL){ /* Динамично начално заделяне на памет за масива */
            return p;
        }
    }
    if(*nm<*sizeAd) { p[(*nm)++]=digit;}
    else {
        *sizeAd+=DELTA;
        tmp=(int *)realloc(p,(*sizeAd)*sizeof(int)); /* разширяване на масива */
        if(tmp!=NULL) {p=tmp; p[(*nm)++]=digit;}
        else {free(p); p=NULL;}
    }
    return p;
}

int main(){
    int *ptr=0; /* Указател към цял тип за организиране на масива */
    FILE *fp=NULL; /* Указател за връзка с файл */
    int i=0,size=0,num=0,digit;
    /* i – индекс; size -размер на масива; num – брой на елементите му; digit – число за вмъкване; */
    if((fp=fopen("test.txt", "r")) == NULL) { /* Отваряне на текстов файл за четене*/
        printf("Cannot open file.\n"); return 2;
    }
    while(fscanf(fp,"%d",&digit)==1) /* Начало на цикъл за четене на числата от файла */ {
        ptr=include(digit,ptr,&size,&num);
        if(ptr==NULL) {
            printf("Memory allocation error!");
            fclose(fp); return 1;
        }
    }
    /* Обработки на масива */
    for(i=0;i<num;i++){
        if(!(i%20)) printf("\n");
        printf("%d",ptr[i]);
    }
    free(ptr);
    fclose(fp);
    return 0;
}
```