Саркисян Г.Ф.

ПРОГРАММИРОВАНИЕ

Часть I: Структурное программирование

ФУНКЦИИ ВЫДЕЛЕНИЯ ПАМЯТИ

ФУНКЦИИ ВЫДЕЛЕНИЯ ПАМЯТИ

Ядром динамического выделения памяти являются функции, объявленные в стандартной библиотеке в заголовочном файле **stdlib.h**

1. **void * malloc (unsigned size)**

1.1. выделяет область памяти размером *size* байтов;

1.2. в случае успеха, возвращает указатель на начало выделенного блока памяти;

1.3. если для выделенной памяти не хватает места, возвращает **NULL**.

ФУНКЦИИ ВЫДЕЛЕНИЯ ПАМЯТИ

2. `void* calloc(unsigned num ,unsigned size)`

2.1 выделяет область памяти размером **`num*size`** байтов;

2.2 в случае успеха, возвращает указатель на начало выделенного блока памяти;

2.3 выделяемая память автоматически инициализируется нулями;

2.4 если для выделенной памяти не хватает места, возвращает **`NULL`**.

ФУНКЦИЯ ПЕРЕОПРЕДЕЛЕНИЯ ВЫДЕЛЕННОЙ ПАМЯТИ

ФУНКЦИЯ ПЕРЕОПРЕДЕЛЕНИЯ ВЫДЕЛЕННОЙ ПАМЯТИ

```
void * realloc (void *ptr ,unsigned size)
```

1. изменяет размер *динамически выделенной памяти*, на которую указывает **ptr** на новый размер – *size* байтов.

Т.е для того чтобы можно было изменить размер памяти его необходимо обязательно динамически выделить с помощь функции **malloc** или функции **calloc**;

ФУНКЦИЯ ПЕРЕОПРЕДЕЛЕНИЯ ВЫДЕЛЕННОЙ ПАМЯТИ

2. если указатель **ptr** не является значением, которое ранее было определено функциями **malloc** или **calloc**, то поведение функции **realloc** не определено;

3. при изменении размера выделяется новая память, а не меняется размер ранее выделенной памяти, следовательно необходимо переадресовывать указатель;

ФУНКЦИЯ ПЕРЕОПРЕДЕЛЕНИЯ ВЫДЕЛЕННОЙ ПАМЯТИ

4. если размер **`size`** больше, чем размер ранее существовавшего блока, то новое, неинициализированное пространство будет выделено в конце блока и предыдущее содержимое пространства сохраняется;

5. если размер **`size`** меньше, чем размер ранее существовавшего блока, то в новом пространстве не будет содержать информацию находящуюся в конце предыдущего блока;

ФУНКЦИЯ ПЕРЕОПРЕДЕЛЕНИЯ ВЫДЕЛЕННОЙ ПАМЯТИ

6. если для переопределения памяти не хватает места, то функция возвращает **NULL** и содержимое пространства на которое указывает **ptr**, остается нетронутым;

7. если значение **size = 0**, а **ptr** не ноль, то функция **realloc** действует как функция **free**.

ФУНКЦИЯ ОСВОБОЖДЕНИЯ ПАМЯТИ

ФУНКЦИЯ ОСВОБОЖДЕНИЯ ПАМЯТИ

```
void  free (void *ptr);
```

Функция *освобождает* область памяти, ранее выделенной при помощи функции **malloc**, **calloc** или **realloc** на которую указывает **ptr**;

1. если **ptr**– **NULL**, то **free** ничего не выполняет.
2. если **ptr** не является указателем, проинициализированным ранее одной из функций выделения памяти, то поведение функции не определено;
3. функция **free** не располагает средствами передачи ошибки, возможно возникающей при ее

ФУНКЦИЯ ОСВОБОЖДЕНИЯ ПАМЯТИ

```
void  free (void *ptr);
```

Функция *освобождает* область памяти, ранее выделенной при помощи функции **malloc**, **calloc** или **realloc** на которую указывает **ptr**;

1. если **ptr** не является указателем, проинициализированным ранее одной из функций выделения памяти, то поведение функции не определено;

ФУНКЦИЯ ОСВОБОЖДЕНИЯ ПАМЯТИ

2. функция **free** не располагает средствами передачи ошибки, возможно возникающей при ее выполнении.

3. если **ptr**– **NULL**, то **free** ничего не выполняет.

ПРИМЕРЫ ВЫДЕЛЕНИЯ ПАМЯТИ

ПРИМЕРЫ ВЫДЕЛЕНИЯ ПАМЯТИ

Для того чтобы динамически выделить память необходимо:

1.Определить указатель, на тот тип для которого выделяется память ***Type*ptr;***

2. Выделить память либо с помощью функции *malloc* либо с помощью *calloc*

2.1 *с помощью функции* ***malloc***

```
ptr = (Type*) malloc( n * sizeof (Type));
```

ПРИМЕРЫ ВЫДЕЛЕНИЯ ПАМЯТИ

2.2 *с помощью функции* **`calloc`**

```
ptr = (Type*) calloc ( n , sizeof (Type));
```

3. Проверка на выделения памяти

```
if ( !ptr ){
  puts(" Not enough memory ");
  return;
}
```

4. После использования освободить:

```
free ( ptr );
```

ПРИМЕРЫ ВЫДЕЛЕНИЯ ПАМЯТИ

Одномерный массив

```
int*arr;
int n i;
printf("Enter the size of array: ");
scanf("%d", &n);
if(n <= 0) { puts("Errors"); return;}

arr = (int*) malloc ( n * sizeof(int));
if( !arr ) {
  printf("Not enough memory \n");
  exit(1);
}
. . . . . .
  free(arr);
```

ПРИМЕРЫ ВЫДЕЛЕНИЯ ПАМЯТИ

Двумерный массив

```
int**arr;
int row, col, i, j;
printf("Enter row: ");  scanf("%d", &row);
printf("Enter col: ");  scanf("%d", &col);
if(row <= 0 || col <= 0) {
    puts("Errors"); return;
}
```

ПРИМЕРЫ ВЫДЕЛЕНИЯ ПАМЯТИ

```
arr = (int**) malloc ( row * sizeof(int*));
for(i = 0 ; i < row; ++i)
  arr[i] = (int*) malloc(col*sizeof(int));
 if( !arr ) {
   printf("Not enough memory \n");
   exit(1);
}
. . . . . .
 for(i = 0 ; i < row; ++i)    free(arr[i]);
    free(arr);
```

ПРИМЕРЫ ВЫДЕЛЕНИЯ ПАМЯТИ

```
char x[100],  *ptr;
int n ,i, j;
printf("Введите строку: ");
gets(x);
n = strlen(x);
ptr =(char *)malloc((n+1)*sizeof(char));
if( ! ptr ){
  printf("Нет места!!!\n");
  exit(1);
}
```

ПРИМЕРЫ ВЫДЕЛЕНИЯ ПАМЯТИ

```
for( j = 0, i= n-1; i >= 0; --i, ++j)
    ptr[j] = x[i];

ptr[n] = '\0';

if( !strcmpi( x , ptr ) ) puts("Da");
else puts("Net");

free(ptr);
```

СПАСИБО ЗА ВНИМАНИЕ!

ВОПРОСЫ ?

ДИНАМИЧЕСКОЕ РАСПРЕДЕЛЕНИЕ ПАМЯТИ

Автор: Саркисян Гаяне Феликсовна