Лекция 7

Алгоритмические языки и программирование

Часть 1

Динамическое выделение памяти Что это значит?

- Это значит то, что при динамическом выделении памяти, память резервируется не на этапе компиляции, а на этапе выполнения программы. И это дает нам возможность выделять память более эффективно, в основном это касается массивов. С динамическим выделением память, нам нет необходимости заранее задавать размер массива, тем более, что не всегда известно, какой размер должен быть у массива.

Динамическое выделение памяти

Очень часто возникают задачи обработки массивов данных, размерность которых заранее неизвестна. В этом случае возможно использование одного из двух подходов:

- *выделение памяти под статический массив*, содержащий максимально возможное число элементов, однако в этом случае память расходуется не рационально;
- *динамическое выделение памяти* для хранение массива данных.

Динамическое выделение памяти

Для динамического выделение памяти необходимо подключить следующие библиотеки:

- **stdlib.h**
- **malloc.h**

Для динамического выделения памяти используются функции:

- **malloc**
- **calloc**
- **realloc**

Далее мы рассмотрим каждую функцию подробнее.

Функция malloc()

```
void * malloc( size_t sizemem );
```

Функция **malloc** выделяет блок памяти,
размером **sizemem** байт, и возвращает указатель на
начало блока.

Возвращаемое значение:
Указатель на выделенный блок памяти. Тип данных на
который ссылается указатель всегда **void*,** поэтому это тип
данных может быть приведен к желаемому типу данных.
Если функции не удалось выделить требуемый блок
памяти, возвращается нулевой указатель.

Функция realloc()

```
void * realloc(void * ptrmem, size_t size);
```

Функция **realloc** выполняет перераспределение блоков памяти.

Размер блока памяти, на который ссылается параметр **ptrmem** изменяется на **size** байтов. Блок памяти может уменьшаться или увеличиваться в размере.

Возвращаемое значение:

Указатель на перераспределенный блок памяти, который может быть либо таким же, как аргумент **ptrmem** или ссылаться на новое место. Тип данных возвращаемого значения всегда **void***, который может быть приведен к любому другому. Если функции не удалось выделить требуемый блок

Функция calloc()

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

Функция **calloc** выделяет блок памяти для массива размером — **num** элементов, каждый из которых занимает **size** байт, и инициализирует все свои биты в нулями.

В результате выделяется блок памяти размером **number * size** байт, причём весь блок заполнен нулями.

Возвращаемое значение:

Указатель на выделенный блок памяти. Тип данных на который ссылается указатель всегда **void***, поэтому это тип данных может быть приведен к желаемому типом данных.

Если функции не удалось выделить требуемый блок памяти, возвращается нулевой указатель.

Функция free()

```
void free(void * ptrmem);
```

Функция **free** освобождает место в памяти. Блок памяти, ранее выделенный с помощью вызова **malloc**, **calloc** или **realloc** освобождается. То есть освобожденная память может дальше использоваться программами или ОС.

Обратите внимание, что эта функция оставляет значение **ptrmem** неизменным, следовательно, он по-прежнему указывает на тот же блок памяти, а не на нулевой указатель.

Возвращаемое значение:

Функция не имеет возвращаемое значение

Пример malloc()

```c
#include <stdio.h>
#include <malloc.h>
#include <stdlib.h>
#include <locale.h>
int main() {
  int *a;  // указатель на массив
  int i, n;
  setlocale(LC_ALL,"rus");
  printf("Введите размер массива: ");
  scanf("%d", &n);
  a = (int*)malloc(n * sizeof(int));  // Выделение памяти
  for (i = 0; i<n; i++) {  // Ввод элементов массива
    printf("a[%d] = ", i);
    scanf("%d", &a[i]);
}
  for (i = 0; i<n; i++)  // Вывод элементов массива
    printf("%d ", a[i]);
  free(a);
  return 0;
}
```

Пример realloc()

```c
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
int main()
{
  int input, ix;
  int counter = 0;     // счетчик введенных чисел
  int *values = NULL;
  int *many_numbers;
  setlocale(LC_ALL, "rus");
  do {
     printf("Введите целое значение (0 - выход): ");
     scanf("%d", &input);
     counter++;
     many_numbers = (int*) realloc (values, counter * sizeof(int)); // при добавлении нового
     if (many_numbers != NULL){                                      увеличиваем м
       values = many_numbers;
       values[counter - 1] = input;  // добавить к массиву только что введённое число
     }
     else{
       free (values); // удалить массив
       printf("Ошибка перевыделения памяти!/n");
       exit (1);      // завершить работу программы
     }
  } while (input != 0); // пока не введён 0

  printf("Введенные числа: ");
  for (ix = 0; ix < counter; ix++)
    printf("%d ", values[ix]);
  free (values); // удалить массив
  return 0;
}
```

Пример calloc()

```c
#include <stdio.h>
#include <malloc.h>
#include <stdlib.h>
#include <locale.h>
int main() {
  int *a;  // указатель на массив
  int i, n;
  setlocale(LC_ALL,"rus");
  printf("Введите размер массива: ");
  scanf("%d", &n);
  a = (int*)calloc(n, sizeof(int));  // Выделение памяти
  for (i = 0; i<n; i++) {  // Ввод элементов массива
    printf("a[%d] = ", i);
    scanf("%d", &a[i]);
}
  for (i = 0; i<n; i++)  // Вывод элементов массива
    printf("%d ", a[i]);
  free(a);
  return 0;
}
```

Лабораторные работы

Динамическое выделение памяти

Напишите программу, которая создает строку длиной указанный пользователем и заполняет её символами английского алфавита. Возможная длина этой строки ограничена только количеством свободной памяти в системе, которую malloc, calloc, realloc может выделить.