Практическая работа 4

При объявлении статического массива, его размер должен быть определён константой. В процессе выполнения программы размер такого массива не может быть изменен.

В процессе решения некоторых задач возможны ситуации, когда объем обрабатываемых данных увеличивается (например, если необходимо объединить два массива данных в один).

Для обработки таких случаев следует уметь выделять память для хранения данных динамически – т.е. непосредственно во время выполнения программы.

Для выделения памяти существует ряд функций. Рассмотрим некоторые из библиотеки `stdlib.h`.

`void *malloc(size_t size);` - выделяет блок памяти. Тип `size_t` есть синоним типа `unsigned int`. `size` – количество байт, которое нужно выделить. `malloc` возвращает указатель типа `void` (указатель неопределенного типа) на выделенную область памяти, либо `NULL` если отсутствует необходимый объем свободной памяти. Для указателя на объект неопределённого типа отсутствует информация о размерах и внутренней структуре адресуемого участка памяти. Из-за этого не могут быть определены какие-либо операции для преобразования значений. Объектам типа указатель на объект неопределённого типа в качестве значений разрешается присваивать значения лишь в сочетании с операцией явного преобразования типа. В этом случае указатель на объект неопределённого типа становится обычным указателем на объект какого-либо конкретного типа.

`void *calloc(size_t num, size_t size);` - выделяет массив ячеек памяти, инициализированных `0`. `num` – количество элементов, `size` – размер одного элемента. `calloc` возвращает указатель типа `void` на выделенную область памяти.

`void *realloc(void *memblock, size_t size);` - перераспределяет память. `memblock` – указатель на изначально выделенную память. `size` - новый размер памяти в байтах. `realloc` возвращает указатель тип `void` на перераспределенный (и, возможно, перемещенный) блок памяти. Если имеется недостаток памяти, чтобы расширить блок до указанного размера, исходный блок памяти не изменяется, а функция возвращает `NULL`. Если исходный блок памяти имеет размер `0`, функция возвращает `NULL`, память не выделяется.

`void free(void *memblock);` - освобождает блок памяти. `memblock` – указатель на изначально выделенную память. Функция освобождает блок памяти, изначально выделенный функциями `calloc`, `malloc`, или `realloc`.

Работа с динамически выделенной памятью обычно происходит в следующей последовательности:

1) объявить указатель на тип данных, с которым предстоит работать. Например:

```
int *n;
```

2) выделить память необходимого размера необходимой функцией. Например:

```
n = (int *)malloc(sizeof(int) * 1000);
```

Параметр функции `malloc` – размер блока памяти, который нужно выделить. Если необходимо создать массив из 1000 элементов типа `int`, то необходимо выделить 1000 ячеек памяти, размер каждой из которых подходит для данных типа `int`. Оператор `sizeof` позволяет вычислить размер, который занимают данные заданного типа; после умножения на 1000 получится размер, необходимый для 1000 элементов заданного типа. Функция возвращает указатель типа `void *`, который необходимо привести с `int *` для последующего использования. Вызов

`n = (int *)malloc(sizeof(int) * 1000);` полностью аналогичен вызову `n = (int *)calloc(sizeof(int), 1000);`

3) выполнить необходимые действия;

4) высвободить выделенную память:

```
free(n);
```

Рассмотрим простую программу, которая запрашивает у пользователя 2 массива, затем присоединяет первый ко второму.

Создайте проект S4_1.

Подключите необходимые библиотеки.

Вот функция `_tmain`. Тщательно разберитесь с каждым символом программы.

```
int _tmain(int argc, _TCHAR* argv[])
{
	setlocale(LC_ALL, ".1251");
	// объявление указателей на будущие массивы
	int *arr1, *arr2;
	// объявление размеров массивов
	int arr1_size, arr2_size;
	// запрос размера массивов
	printf("Введите размер первого массива: ");
	scanf("%d", &arr1_size);
	printf("\nВведите размер второго массива: ");
	scanf("%d", &arr2_size);
	// выделение памяти для первого массива с заданным пользователем размером
	arr1 = (int *)malloc(sizeof(int) * arr1_size);
	// выделение памяти для второго массива с заданным пользователем размером
	// при помощи другой функции (для развлечения)
	arr2 = (int *)calloc(sizeof(int), arr2_size);
	// если выделение памяти завершилось неудачно, завершить работу
	if (arr1 == NULL || arr2 == NULL) return;
	// заполнение первого массива значениями, вводимыми пользователем
	// (напишите функцию самостоятельно)
	fillArray(arr1, arr1_size);
	// заполнение второго массива значениями, вводимыми пользователем
	fillArray(arr2, arr2_size);
	// вывод значений элементов массива на экран
	printArray(arr1, arr1_size);
	// вывод значений элементов массива на экран
	printArray(arr2, arr2_size);
	// перераспределение памяти: по адресу, по которому располагается arr1
	// дополнительновыделяется столько памяти,
	// сколько занимает arr2. Т.е. итоговый размер - arr1_size+arr2_size.
	// Выделенная память записывается по адресу первого массива
	arr1 = (int *)realloc(arr1, sizeof(int) * (arr1_size + arr2_size));
	// копирование данныех из второго массива в расширенный первый
	for (int i = arr1_size; i <= arr1_size + arr2_size - 1; i++) {
		arr1[i] = arr2[i - arr1_size];
	}
	// вывод первого массива, дополненного данными из второго
	printArray(arr1, arr1_size + arr2_size);
	// освобождение памяти
	free(arr1);
	free(arr2);
	return 0;
}
```

Самостоятельно разработайте функции fillArray и printArray.

Задание для самостоятельной работы.

Выполняется в проекте S4_2.

В программе должно быть такое меню, которое позволяет выполнить соответствующие действия с динамической памятью:

1. Задать размер массива
2. Заполнить массив
3. Изменить размер массива
4. Вывести содержимое массива

5. Выйти

Операция 1 позволяет выделить новую память для массива. Следует очищать память, если память уже была выделена.

Операция 2 позволяет заполнить массив значениями.

Операция 3 позволяет перерасределить память для существующего массива.