Информатика

Кочанов Марк

6 октября 2018 г.

МИФИ

Динамическая память

Выделение памяти

```c
#include <stdlib.h>      // malloc, free, exit
#include <stddef.h> // NULL
#include <stdio.h> // printf, NULL
int n = 10;
char *buffer;
buffer = (char*) malloc(n+1);
if (buffer == NULL) exit(1);
for (int i = 0; i < n; i++)
    buffer[i] = rand()%26 + 'a';
buffer[n] = '\0';
printf ("Random string: %s\n", buffer);
free (buffer);
```

Функция malloc()

- Возвращает указатель на область памяти, которая ничем не инициализирована (лежит мусор)[1]
- Возвращает нетипизированный указатель (void $*$)
- В языке C++ неявное приведение указателей запрещено, необходимо явное приведение типов
- Необходимо вручную расчитывать количество байтов, необходимых для хранения данных
- В случае ошибки выделения памяти malloc() возвращает нулевой указатель (NULL)
- Программист отвечает за освобождение памяти, в отличие от автоматических (локальных) переменных
- Имеются накладные расходы на выделение/освобождение памяти по сравнению с автоматическими переменными

[1] Функция calloc () позволяет выделить память и заполнить её нулями.

Использование

```c
#include <stdlib.h>      // malloc, free, exit
#include <stddef.h> // NULL
#include <stdio.h> // printf, NULL
int n = 10;
char *buffer;
buffer = (char*) malloc(n+1);
if (buffer == NULL) exit(1);
for (int i = 0; i < n; i++)
    buffer[i] = rand()%26 + 'a';
buffer[n] = '\0';
printf ("Random string: %s\n", buffer);
free (buffer);
```

- sizeof возвращает число байтов, необходимое для хранения указанного типа данных
- Может быть использована для встроенных типов данных, для указателей, для пользовательсих типов данных (struct)
- sizeof является встроенным оператором языка
- Имеет альтернативный синтаксис: sizeof int
- Возвращает тип данных size_t (unsigned integer), определенный в заголовочном файле stddef.h

Двумерный массив (массив указателей на массивы)

```
1  int **c;
2  int nrows = 100, ncolumns = 200;
3
4  if (( c = malloc(nrows * sizeof(int *))) == NULL )
5      exit(1);
6
7  for (int i = 0; i < nrows; i++ ) {
8      if (( c[i] = malloc(ncolumns * sizeof(int))) == NULL )
9          exit(2);
10 }
11
12 // c[45][30] = 10
13
14 for (int i = 0; i < nrows; i++ )
15     free(c[i]);
16 free(c);
```

Двумерный массив (версия 2)

Код приведен в учебных целях, не следует его использовать когда-либо.

```
1  int **c = malloc(nrows * sizeof(int *)
2                   + nrows * ncolumns * sizeof(int));
3  int *offs = &array[nrows];
4  for (int i = 0; i < nrows; i++, offs += ncolumns) {
5      c[i] = offs;
6  }
7
8  // c[10][20] = ...
9
10 free(c);
```

Задачи

Длинная арифметика

Написать функции для сложения и вычитания двух чисел, для хранения которых недостаточно встроенных типов данных.

Функция сложения принимает на вход два указателя типа char* (массивы из char) указывающие на два исходных числа. Третий указатель типа char** (указатель на массив из char) соответствует результату сложения. Четвертый аргумент функции является указателем типа int*, через который функция возвращает информацию о длине результирующего числа. Функция внутри себя выделяет память для хранения результата вычислений и производит необходимые вычисления.

Функция вычитания двух чисел имеет такую же сигнатуру и проверяет внутри себя, что первое число не меньше второго. В противном случае она завершается с ошибкой.

Также необходимо написать функция для вывода в консоль числа.

Прототипы

```
1  // Calculate c=a+b and store length of c in len
2  void sum(int* a, int* b, int** c, int* len);
3
4  // Calculate c=a-b and store length of c in len
5  void subtract(int* a, int* b, int** c, int* len);
```