Основы C/C++ – Домашнее задание 14

Пеганов Антон
peganoff2@mail.ru

30 марта 2020 г.

Задача 1

Напишите функцию `binary_search_rec()`, выполняющую двоичный поиск с помощью рекурсии.

Задача 2

Вам нужно написать функцию `print_array_6()`, которая выводит на экран последовательность из шести целых чисел. Числа выводятся через пробел, в конце символ конца строки.

Для проверки будет использоваться следующий код. Вы можете использовать его для отладки.

Листинг 1: Код, вызывающий функцию `print_array_6()`.

```
#include <iostream>
using std::cin;
using std::cout;
using std::endl;

void print_array_6(int* p);

int main() {
    int a[6];
    for (int i = 0; i < 6; i++)
        cin >> a[i];
    print_array_6(a);
    return 0;
}
```

Формат входных данных

Последовательность целых чисел.

Формат выходных данных

Ровно та же последовательность целых чисел.

Примеры

Ввод	Вывод
1 2 3 4 5 6	1 2 3 4 5 6

Задача 3

Вам нужно написать две функции.
Первая, `now_get_me_some_bytes()`, выделяет в куче (heap) место под массив из `n` элементов типа `int` и возвращает указатель на начало массива. В качестве единственного аргумента она принимает размер массива. Гарантируется, что он будет не меньше единицы. Вторая, `now_free_some_bytes()`, корректно очищает эту память. В качестве единственного аргумента она принимает указатель на память, которая была выделена при помощи функции `now_get_me_some_bytes()`.

Листинг 2: Код, вызывающий функции `now_get_me_some_bytes()` и `now_free_some_bytes()`.

```
#include <iostream>
using std::cin;
using std::cout;
using std::endl;

int* now_get_me_some_bytes(unsigned int n);
void now_free_some_bytes(int* p);

int main() {
    unsigned int n;
    cin >> n;
    int *a = now_get_me_some_bytes(n);
    for (int i = 0; i < n; i++)
        cin >> a[i];
    int s = 0;
    for (int i = 0; i < n; i++)
        s += a[i];
    // `int` variables are interpreted as `false`
    // if they are equal to zero and as `true` otherwise.
    if (s % 2)
        cout << "NO" << endl;
    else
        cout << "YES" << endl;
    now_free_some_bytes(a);
    return 0;
}
```

Обратите внимание, что ответственность за корректный вызов этих функций в нужном порядке ложится на проверяющую программу. Это простая задача, вы можете не писать проверку корректности аргументов и прочую "защиту от дурака". Сосредоточьтесь на корректной работе с памятью.

Для проверки будет используется код из Листинга 2. Вы можете использовать его для отладки.

Он считывает размер массива целых чисел с экрана, с помощью вашей функции выделяет под него память, потом считывает с экрана элементы массива, суммирует их и определяет, четное ли получилось число. Вообще-то для решения такой задачи массивы не требуются - здесь этот алгоритм исключительно для того, чтобы проверить, правильно ли выделилась память.

Формат входных данных
Последовательность целых чисел.

Формат выходных данных
Ровно та же последовательность целых чисел.

Примеры

Ввод	Вывод
1 1 2 3 4 5 6	YES
1 12345	YES

Задача 4

Вам нужно написать функцию `my_personal_swap()`, которая меняет местами значения двух переменных типа `int`. В качестве аргументов она принимает два указателя, которые хранят адреса этих переменных. Обратите внимание, что если вам передали некорректные указатели (`NULL`), функция должна это проверять и не пытаться эти указатели разыменовывать. В этом случае она просто ничего не делает, на экран выводить тоже ничего не нужно.
Для проверки будет использоваться следующий код. Вы можете использовать его для отладки.

Листинг 3: Код, вызывающий функцию `my_personal_swap()`.

```
#include <iostream>
using std::cin;
using std::cout;
using std::endl;

void my_personal_swap(int* a, int* b);

int main() {
    int a, b;
    cin >> a >> b;
    my_personal_swap(&a, &b);
    cout << a << " " << b << endl;
    my_personal_swap(&a, NULL);
    my_personal_swap(NULL, &b);
    my_personal_swap(NULL, NULL);
    return 0;
}
```

Формат входных данных
Два целых числа.

Формат выходных данных
Те же числа в обратном порядке.

Примеры

Ввод	Вывод
1 2	2 1

Задача 5

У вас есть функция `int do_some_awesome_work(int* a, int* b)`. Вы не знаете, что она делает, но вам нужно ее использовать. Для этого вы должны считать с экрана два числа, правильно вызвать эту функцию и вывести на экран то, что она вернула.

Для отладки можете написать себе любую реализацию этой функции.

Формат входных данных
Два целых числа.

Формат выходных данных
Те же числа в обратном порядке.

Примеры

Ввод	Вывод
5 6	5
-1 -3	-3

Задача 6

Вам нужно написать функцию `int* my_slightly_dumb_reallocation(int* source, unsigned int n_old, unsigned int n_new)`, которая меняет размер памяти, выделенной под массив целых чисел. Она должна выделить новый кусок памяти нужного размера, перенести туда данные из старого массива, очистить память по старому адресу и вернуть указатель на новый массив.

В качестве аргументов она принимает:

1. Указатель на массив, память для которого была выделена в куче (heap) при помощи оператора `new[]`. Обратите внимание, что если вам передали некорректный указатель (`NULL`), функция должна это проверять и не пытаться этот указатель разыменовывать или очищать память по этому адресу.

2. Количество элементов массива на момент вызова функции.

3. Новое количество элементов. Если оно меньше, чем старое, то вы копируете только те элементы, которые влезают, и ничего не выводите на экран. Часть данных потеряется,

но это на совести того, кто вашу функцию вызывал. В реальной жизни в таком случае хорошо бы печатать предупреждение.

Для проверки будет использоваться следующий код. Вы можете использовать его для отладки.

Листинг 4: Код, вызывающий функцию `int* my_slightly_dumb_reallocation(int* source, unsigned int n_old, unsigned int n_new)`

```
#include <iostream>
using std::cin;
using std::cout;
using std::endl;

int* my_slightly_dumb_reallocation(
        int* source, unsigned int n_old, unsigned int n_new);

int main() {
    unsigned int n, i;
    cin >> n;
    int *a = my_slightly_dumb_reallocation(NULL, 0, n / 2);
    for (i = 0; i < n / 2; i++)
        cin >> a[i];
    a = my_slightly_dumb_reallocation(a, n / 2, n);
    for (; i < n; i++)
        cin >> a[i];
    int sum = 0;
    for (i = 0; i < n; i++)
        sum += a[i];
    cout << sum << endl;
    a = my_slightly_dumb_reallocation(a, n, n / 2);
    a = my_slightly_dumb_reallocation(a, n / 2, 0);
    a = my_slightly_dumb_reallocation(a, 0, 0);
    return 0;
}
```

Формат входных данных

Первое число `n` — количество элементов. Последовательность из `n` целых чисел.

Формат выходных данных

Те же числа в обратном порядке.

Примеры

Ввод	Вывод
5 1 2 3 4 5	15

Задача 7

Вам нужно написать функцию `void my_slightly_less_dumb_reallocation(int** source, unsigned int n_old, unsigned int n_new)`, которая меняет размер памяти, выделенной

под массив целых чисел. Она должна выделить новый кусок памяти нужного размера, перенести туда данные из старого массива, очистить память по старому адресу и изменить указатель на массив.

В качестве аргументов она принимает:

1. Указатель на указатель на массив, память для которого была выделена в куче (heap) при помощи оператора `new[]`. Обратите внимание, что если вам передали некорректный указатель (`NULL`) любого уровня, функция должна это проверять и не пытаться этот указатель разыменовывать или очищать память по этому адресу.
2. Количество элементов массива на момент вызова функции.
3. Новое количество элементов. Если оно меньше, чем старое, то вы копируете только те элементы, которые влезают, и ничего не выводите на экран. Часть данных потеряется, но это на совести того, кто вашу функцию вызывал.

Для проверки будет использоваться следующий код. Вы можете использовать его для отладки.

Листинг 5: Код, вызывающий функцию `int* my_slightly_less_dumb_reallocation(int* source, unsigned int n_old, unsigned int n_new)`

```
#include <iostream>
using std::cin;
using std::cout;
using std::endl;

void my_slightly_less_dumb_reallocation(
        int** source, unsigned int n_old, unsigned int n_new);

int main() {
    unsigned int n, i;
    cin >> n;
    int *a = NULL;
    my_slightly_less_dumb_reallocation(&a, 0, n / 2);
    for (i = 0; i < n / 2; i++)
        cin >> a[i];
    my_slightly_less_dumb_reallocation(&a, n / 2, n);
    for (; i < n; i++)
        cin >> a[i];
    int sum = 0;
    for (i = 0; i < n; i++)
        sum += a[i];
    cout << sum << endl;
    my_slightly_less_dumb_reallocation(&a, n, n / 2);
    my_slightly_less_dumb_reallocation(&a, n / 2, 0);
    my_slightly_less_dumb_reallocation(&a, 0, 0);
    return 0;
}
```

Формат входных данных

Первое число `n` — количество элементов. Последовательность из `n` целых чисел.

Формат выходных данных
Те же числа в обратном порядке.

Примеры

Ввод	Вывод
5 1 2 3 4 5	15

Задача 8

Вам нужно написать функцию `unsigned int count_total_mice_amount(Cat* cats, unsigned int n)`, которая считает общее количество мышей, пойманных котами — возвращает сумму полей `mice_caught` элементов массива структур `Cat`. В качестве аргумента она принимает указатель на начало массива и количество элементов в нем.

Для проверки будет использоваться следующий код. Вы можете использовать его для отладки.

Листинг 6: Код, вызывающий функцию `count_total_mice_amount(Cat* cats, unsigned int n)`

```
#include <iostream>
using std::cin;
using std::cout;
using std::endl;

struct Cat {
    char name[20];
    unsigned int id;
    double weight, length;
    unsigned int mice_caught;
};

unsigned int count_total_mice_amount(Cat* cats, unsigned int n);

int main() {
    unsigned int n;
    cin >> n;
    Cat *a = new Cat[n];
    for (int i = 0; i < n; i++) {
        cin >> a[i].name >> a[i].weight >> a[i].length
            >> a[i].mice_caught;
        a[i].id = i;
    }
    cout << count_total_mice_amount(a, n) << endl;
    delete[] a;
    return 0;
}
```

Формат входных данных
Число `n` — количество котов. Вся необходимая информация о каждом коте — имя, вес, длина и количество мышей.

Формат выходных данных
Сумма всех пойманных мышей.

Примеры

Ввод	Вывод
5 Lanfear 1.3 23.6 1000 Annoura 2.5 1.6 20 Atuan 1.6 0.6397 15 Leane 1.7 0.684 3 Liandrin 2.6 0.257 165	1203

Задача 9

Вам нужно написать две функции.

1. `Cat* get_home_for_a_cats_pride(unsigned int n)` выделяет память под массив из `n` котов — структур `Cat`. В качестве аргумента она принимает нужное количество элементов, а возвращает указатель на начало выделеной памяти.

2. `void clear_home_of_a_cats_pride(Cat *cats, unsigned int n)` корректно очищает эту память. В качестве единственного аргумента она принимает указатель на память, которая была выделена при помощи функции `get_home_for_a_cats_pride()`.

Обратите внимание, что вам также надо выделить память под поле name. Считаем, что имя кота не длиннее 10 символов.

Ответственность за корректный вызов этих функций в нужном порядке ложится на проверяющую программу. Это простая задача, вы можете не писать проверку корректности аргументов и прочую "защиту от дурака". Сосредоточьтесь на корректной работе с памятью.

Для проверки будет использоваться следующий код. Вы можете использовать его для отладки.

Листинг 7: Код, вызывающий функции `get_home_for_a_cats_pride()` и `clear_home_of_a_cats_pride()`

```
#include <iostream>
using std::cin;
using std::cout;
using std::endl;

struct Cat {
    char *name;
    unsigned int id;
    double weight, length;
    unsigned int mice_caught;
};

Cat* get_home_for_a_cats_pride(unsigned int n);
void clear_home_of_a_cats_pride(Cat *cats, unsigned int n);

int main() {
    unsigned int n;
    cin >> n;
    Cat *a = get_home_for_a_cats_pride(n);
    for (int i = 0; i < n; i++) {
        cin >> a[i].name >> a[i].weight >> a[i].length
            >> a[i].mice_caught;
        a[i].id = i;
    }
    for (int i = 0; i < n; i++)
        cout << a[i].name << " ";
    cout << endl;
    clear_home_of_a_cats_pride(a, n);
    return 0;
}
```

Формат входных данных

Число `n` — количество котов. Вся необходимая информация о каждом коте — имя, вес, длина и количество мышей.

Формат выходных данных

Имена котов в строку без пробелов.

Примеры

Ввод	Вывод
5 Lanfear 1.3 23.6 1000 Annoura 2.5 1.6 20 Atuan 1.6 0.6397 15 Leane 1.7 0.684 3 Liandrin 2.6 0.257 165	Lanfear Annoura Atuan Leane Liandrin

Что такое ассоциативный массив и как его используют?

Ассоциативный массив — структура данных, элементами которой являются пары ключ-значение и которая позволяет получить значение элемента по ключу. Например, ассоциативным массивом является англо-русский словарь. Русский перевод мы получаем, когда находим в словаре ключ — английское слово.

В ассоциативном массиве каждый ключ может встречаться только в одном экземпляре, также как в обычном массиве не может быть элементов с одинаковыми индексами.

Ассоциативные массивы применяются повсеместно, например в базах данных для поиска объектов с некоторым значением признака.

В C++ ассоциативные массивы представлены типами `map` (карта) и `unordered_map` (неупорядоченная карта) (`unordered_map` — начиная с версии C++11).

Следующую задачу удобно решать с помощью `map`. Некоторые возможности типа демонстрируются в файле *cpp/hw11/map_ example.cpp*.

Задача 10

Решите задачу 987A.

Задача 11

Решите задачу 897A