ТЕМА: ВВЕДЕНИЕ В ЯЗЫК ПРОГРАММИРОВАНИЯ «C++»

Домашнее задание 2

ЗАДАНИЕ 1

Написать программу, копирующую последовательно элементы одного массива размером 10 элементов в 2 массива размером 5 элементов каждый. *Элементы первоначального массива можно сгенерировать в произвольном диапазоне.*

Подсказка 1 — Если у Вас возникли сложности с выполнением данного задания, нажмите кнопку «Подсказка». С полным решением задания вы сможете ознакомиться, нажав на кнопку «Решение» — Решение 1

ЗАДАНИЕ 2

Напишите программу, которая выполняет поэлементную сумму двух массивов и результат заносит в третий массив. *Элементы массивов можно сгенерировать в произвольном диапазоне.*

Подсказка 2 — Если у Вас возникли сложности с выполнением данного задания, нажмите кнопку «Подсказка». С полным решением задания вы сможете ознакомиться, нажав на кнопку «Решение» — Решение 2

ЗАДАНИЕ 3

Пользователь вводит данные о своих расходах в долларах за неделю (каждый день). Написать программу, которая вычисляет:

- среднюю (за неделю) потраченную сумму;
- общую сумму, потраченную пользователем за неделю;
- количество дней и их названия (например, «вторник»), когда сумма расхода (в день) превысила 100 долларов.

ЗАДАНИЕ 4

Известны данные курса валют (курс доллара по отношению к евро) за все месяцы года и начисленные проценты за каждый месяц на депозитном счету в евро. Все данные вводятся пользователем, в том числе и сумма на депозите в евро.

Написать программу, которая по запросу пользователя (номер месяца) выводит в консоль размер допустимой суммы, которую он может обналичить, при условии, что у него долларовая карта, обналичить можно не более 50% от начисленной суммы в том случае, если начисленная сумма в этот месяц составляет не менее 500$.

Финальные задания рассчитаны на самостоятельное решение. Здесь не будет подсказок и готового алгоритма. Вам необходимо применить все практические навыки, полученные из предыдущих заданий.

ПОДСКАЗКА К ЗАДАНИЮ 1

1. При объявлении массива необходимо указать типа и его размерность.
2. С помощью какой функции в C++ можно сгенерировать случайное число? Необходимо ли подключать дополнительную библиотеку для этого?
3. Последовательное копирование — это поэлементное копирование (вначале первый элемент, затем следующий и т. д.) из одного массива в другой.
4. С помощью какого оператора значение элемента из одного массива может быть скопировано (занесено) в элемент другого массива?
5. Если в первый массив копируется первая часть основного массива, а во второй — вторая, то это значит, что в нулевой элемент первого массива скопируется нулевой элемент с начального массива, а в нулевой элемент второго массива — пятый элемент с начального массива и т. д.
6. Какой длины будет цикл по копированию элементов?

ПОДСКАЗКА К ЗАДАНИЮ 2

1. При объявлении массива необходимо указать типа и его размерность. Размерности всех трех массивов должны быть одинаковы.
2. С помощью какой функции в C++ можно сгенерировать случайное число? Необходимо ли подключать дополнительную библиотеку для этого?
3. Поэлементное суммирование — это последовательное повторение операции суммы над парой элементов (с одинаковыми индексами), новой на каждом следующем шаге.
4. Как связана длина цикла, в котором выполняется операция суммирования и длина массива?

РЕШЕНИЕ ЗАДАНИЯ 1

Описание решения

Вначале необходимо определиться с размерность массивов (максимальным количеством элементов, которое может в нем храниться). В нашем примере длина начального массива — десять, а массивов для копирования — пять.

Так как в массив будут заноситься случайные числа, то необходимо использовать генератор случайных чисел, а именно функции srand() для установки стартового значения генератора и rand(), которая генерирует случайное число в последовательности.

Для того, чтобы при каждом новом запуске программы генерировалось новое число в функцию srand() в качестве параметра нужно передавать системное время. Для получения текущего системного времени будем использовать функцию time(). Чтобы использовать эту функцию, нужно подключить заголовочный файл ctime (или #include "time.h"), а затем инициализировать функцию srand() вызовом функции time(NULL).

Так как нам нужны случайные числа в определенном диапазоне, то следует выполнить преобразование полученного от функции rand() числа в число из указанного диапазона. Это выполняется с помощью следующей формулы:

a + rand() % (b — a),

где a и b — границы нужного диапазона.

Так как в каждый элемент массива нужно заносить новое случайное число, то процесс генерации следует повторять в цикле, длина которого будет равна длине массива.

Если в первый массив (a2) копируется первая часть основного массива (a1), а во второй (a3) — вторая, то это значит, что в нулевой элемент первого массива скопируется нулевой элемент с начального массива, а в нулевой элемент второго массива — пятый элемент с начального массива и т.д.

Таким образом, длина цикла, в котором будет выполняться данное копирование составит пять итераций (половина длина начального массива).

Можно выполнять копирование каждой из половин начального массива в одном цикле, если использовать сдвиг индексов.

Например,

a1[0] копируется в a2[0]
a1[1] копируется в a2[1]
....
a1[5] копируется в a3[0]
a1[6] копируется в a3[1]
....

Разница (сдвиг) индексов элементов, что помещаются в первый и второй массив, составляет пять, т.е. на i-том шаге:

a2[i] получает данные из a1[i],
a3[i] получает данные из a1[i+5]

Решение

1. Подключаем библиотеку (#include»time.h»), которая позволит в дальнейшем использовать встроенную функцию time() для получения текущего системного времени, которое необходимо для старта генератора псевдослучайных чисел.

Выводим строку в консоль с описанием названия программы (по желанию, данная строка кода может быть пропущена), создаем необходимое число переменных и констант.
Четыре целочисленные константы необходимы для хранения длин массивов, начального и конечного значения диапазона генерируемых чисел. В нашем примере длина начального массива будет 10, первого и второго — 5, а числа генерируются в диапазоне от 10 до 20.
Далее объявляем три целочисленных массива, используя созданные ранее целочисленные константы для задания размеров массивов.

```
#include <iostream>
#include"time.h"

using namespace std;

int main()
{
    cout << "Home task #9.1.1\n";

    int const n = 10, m = 5, a = 10, b = 20;
    int mass[n], mass1[m], mass2[m];

    return 0;
}
```

2. Для того, чтобы при каждом новом запуске программы генерировалось новое число используем функцию srand(), а в качестве ее параметра — вызов функции time(NULL). Далее организуем цикл (длительность которого равна

длине массива), внутри которого выполняются операция генерации случайного числа в заданном диапазоне. После генерации выводим полученное число (элемент массива) в консоль.

```
#include <iostream>
#include"time.h"

using namespace std;
int main()
{
    cout << "Home task #9.1.1\n";

    int const n = 10, m = 5, a = 10, b = 20;
    int mass[n], mass1[m], mass2[m];

    srand(time(NULL));

    for (int i = 0; i < n; i++)
    {
        mass[i] = a + rand() % (b - a);
        cout << mass[i] << " ";
    }

    return 0;
}
```

3. Длина цикла, в котором будет выполняться копирование составит пять итераций (половина длина начального массива). Будем выполнять копирование каждой из половин начального массива в одном цикле, используя сдвиг индексов. Разница (сдвиг) индексов элементов, что помещаются

в первый и второй массив, составляет пять.
После выполнения копирования выводим значения из полученных массивов (в столбик).

```
#include <iostream>
#include"time.h"

using namespace std;

int main()
{
    cout << "Home task #9.1.1\n";
    int const n = 10, m = 5, a = 10, b = 20;
    int mass[n], mass1[m], mass2[m];

    srand(time(NULL));

    for (int i = 0; i < n; i++)
    {
        mass[i] = a + rand() % (b - a);
        cout << mass[i] << " ";
    }

    cout << "\n\n";
    cout << "1st  2nd\n";

    for (int i = 0; i < m; i++)
    {
        mass1[i] = mass[i];
        mass2[i] = mass[i+5];
    }

    for (int i = 0; i < m; i++)
    {
```

```
        cout << mass1[i] << " "<< mass2[i]<<"\n";
    }

    return 0;
}
```

Результаты работы программы (в консоли):

```
Home task #9.1.1
15 11 16 14 14 12 12 19 19 14

1st  2nd
15 12
11 12
16 19
14 19
14 14
```

Рисунок 1

РЕШЕНИЕ ЗАДАНИЯ 2

Описание решения

Вначале необходимо определиться с размерность массивов (максимальным количеством элементов, которое может в нем храниться). Если мы выполняем последовательное суммирование элементов двух массивов, то предполагается, что их длины равны и совпадают также с длиной массива для хранения результата. Пусть в нашем примере длина массива — десять.

Так как и в первый, и во второй массив будут заноситься случайные числа, то необходимо использовать генератор случайных чисел, а именно функции srand() для установки стартового значения генератора и rand(), которая генерирует случайное число в последовательности.

Для того, чтобы при каждом новом запуске программы генерировалось новое число в функцию srand() в качестве параметра нужно передавать системное время. Для получения текущего системного времени будем использовать функцию time(). Чтобы использовать эту функцию, нужно подключить заголовочный файл ctime (или #include "time.h"), а затем инициализировать функцию srand() вызовом функции time(NULL).

Так как нам нужны случайные числа в определенном диапазоне, то следует выполнить преобразование полученного от функции rand() числа в число из указанного диапазона. Это выполняется с помощью следующей формулы:

a + rand() % (b — a),

где a и b — границы нужного диапазона.

Т.к. в каждый элемент массива нужно заносить новое случайное число, то процесс генерации следует повторять в цикле, длина которого будет равна длине массива.

В результирующий массив на каждом шаге заноситься результат сложения пары элементов из начальных массивов, т. е. операция суммирования повторяется для каждой новой (следующей) пары и полученный результат заносится в новый (следующий) элемент результирующего массива. Для реализации повторяющихся операций используем циклы.

В связи с тем, что длина первого и второго цикла совпадает, можно объединить процедуры генерации элементов и их суммирования в теле одного цикла.

Решение

1. Подключаем библиотеку (#include»time.h»), которая позволит в дальнейшем использовать встроенную функцию time() для получения текущего системного времени, которое необходимо для старта генератора псевдослучайных чисел. Выводим строку в консоль с описанием названия программы (по желанию, данная строка кода может быть пропущена), создаем необходимое число переменных и констант. Три целочисленные константы необходимы для хранения длины массива, начального и конечного значения диапазона генерируемых чисел. В нашем примере длина массива будет 10, а числа генерируются в диапазоне от 10 до 20.
   Далее объявляем три целочисленных массива, используя созданную ранее целочисленную константу для задания размеров массивов.

```
#include <iostream>
#include"time.h"

using namespace std;

int main()
{
    cout << "Home task #9.1.2\n";
    int const n = 10, a = 10, b = 20;
    int mass1[n], mass2[n], result[n];

    return 0;
}
```

2. Для того, чтобы при каждом новом запуске программы генерировалось новое число используем функцию srand(), а в качестве ее параметра — вызов функции time(NULL). Далее организуем цикл (длительность которого равна длине массива), внутри которого выполняются операция генерации случайного числа в заданном диапазоне.
   В этом же цикле реализуем операцию суммирования пары элементов двух массивов и заносим результат в соответствующий (с таким же индексом) элемент третьего массива. После генерации и сложения выводим сгенерированные элементы и результат их суммирования в консоль.

```
#include <iostream>
#include"time.h"

using namespace std;
```

```
int main()
{
    cout << "Home task #9.1.2\n";
    int const n = 10, a = 10, b = 20;
    int mass1[n], mass2[n], result[n];
    srand(time(NULL));
    cout << "1st + 2nd = result\n";
    for (int i = 0; i < n; i++)
    {
        mass1[i] = a + rand() % (b - a);
        mass2[i] = a + rand() % (b - a);
        result[i] = mass1[i] + mass2[i];

        cout << mass1[i] << " + " << mass2[i];
        cout <<" = "<< result[i]<< "\n";
    }
    return 0;
}
```

Результаты работы программы (в консоли):

```
Home task #9.1.2
1st + 2nd = result
19 + 19 = 38
17 + 15 = 32
14 + 13 = 27
16 + 16 = 32
15 + 16 = 31
15 + 10 = 25
19 + 10 = 29
17 + 12 = 29
13 + 10 = 23
11 + 10 = 21
```

Рисунок 2