|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ** ***ИУК «Информатика и управление»***

**КАФЕДРА** \_\_***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***

**ЛАБОРАТОРНАЯ РАБОТА**

**«Стандарт кодирования»**

**ДИСЦИПЛИНА: «Основы программной инженерии»**

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИУК4-22Б | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Карельский М.К. )  (Подпись) |
| Проверил: | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Амеличев Г.Э. )  (Подпись) |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | |
|  | - Балльная оценка:  - Оценка: | |

Калуга, 2021

**Цель:** формирование практических навыков следования стандарту кодирования при разработке программного обеспечения.

**Вариант №4**

Квадратную матрицу 17x17 заполнить случайными числами из диапазона [-12; 12], вывести ее на экран.

а) К каждому элементу целочисленной матрицы прибавить сумму цифр элемента, стоящего в этой же строке на побочной диагонали.

б) Проверить, содержит строка, номер которой введен с клавиатуры, знакочередующиеся элементы.

в) Указать номер строки и столбца первого нулевого элемента среди элементов матрицы, выделенных чёрным цветом (матрица квадратная).

**Листинг:**

#include <iostream>

#include <ctime>

#include <iomanip>

using namespace std;

enum MenuCodes

{

PRINT\_ARRAY,

TASK\_1,

TASK\_2,

TASK\_3,

EXIT

};

const int MINIMUM\_RANDOM\_NUMBER = -12;

const int MAXIMUM\_RANDOM\_NUMBER = 12;

const int MINIMUM\_ARRAY\_BORDER = 3;

const int MAXIMUM\_ARRAY\_BORDER = 17;

void Menu();

void FillArray(int\*\* array, int arrayBorder, int mode);

void PrintArray(int\*\* array, int arrayBorder);

int SumOfDigits(int number);

int FindSideDiagonalSameRowNumber(int\*\* array, int lengthOfRow, int numberOfRow);

void IncreaseArrayElements(int\*\* array, int arrayBorder);

void CheckSignAlternation(int\*\* array, int arrayBorder);

void FindZeroElement(int\*\* array, int arrayBorder);

int main()

{

setlocale(LC\_ALL, "Russian");

srand(time(0));

Menu();

return 0;

}

void Menu()

{

cout << "Введите размер квадратной матрицы (" << MINIMUM\_ARRAY\_BORDER << '-' << MAXIMUM\_ARRAY\_BORDER << "): ";

int arrayBorder{};

cin >> arrayBorder;

while (arrayBorder < MINIMUM\_ARRAY\_BORDER || arrayBorder > MAXIMUM\_ARRAY\_BORDER)

{

cout << "Неподходящее значение, попробуйте еще раз: ";

cin >> arrayBorder;

}

cout << "\nВыберете способ задания матрицы\n";

cout << "0. Вручную\n";

cout << "1. Случайно\n";

cout << "Выберете команду : ";

int mode{};

cin >> mode;

while (mode != 0 && mode != 1)

{

cout << "Неверная команда, попробуйте еще раз: ";

cin >> mode;

}

cout << '\n';

int\*\* array = new int\* [arrayBorder] {};

FillArray(array, arrayBorder, mode);

int code = -1;

while (code != EXIT)

{

cout << PRINT\_ARRAY << ". Вывести матрицу\n";

cout << TASK\_1 << ". Сложить элементы с суммой цифр элемента побочной диагонали той же строки\n";

cout << TASK\_2 << ". Проверка строки на чередование знаков\n";

cout << TASK\_3 << ". Поиск первого нулевого элемента\n";

cout << EXIT << ". Выход\n";

cout << "Выберете команду: ";

cin >> code;

while (code < PRINT\_ARRAY || code > EXIT)

{

cout << "Неверная команда, попробуйте еще раз: ";

cin >> code;

}

cout << "\n";

switch (code)

{

case PRINT\_ARRAY:

PrintArray(array, arrayBorder);

break;

case TASK\_1:

IncreaseArrayElements(array, arrayBorder);

break;

case TASK\_2:

CheckSignAlternation(array, arrayBorder);

break;

case TASK\_3:

FindZeroElement(array, arrayBorder);

break;

}

}

}

void FillArray(int\*\* array, int arrayBorder, int mode)

{

if (mode == 0)

{

cout << "Задайте матрицу\n";

}

for (int i = 0; i < arrayBorder; i++)

{

array[i] = new int[arrayBorder] {};

for (int j = 0; j < arrayBorder; j++)

{

if (mode == 0)

{

cin >> array[i][j];

}

else

{

array[i][j] = rand() % (MAXIMUM\_RANDOM\_NUMBER - MINIMUM\_RANDOM\_NUMBER + 1) + MINIMUM\_RANDOM\_NUMBER;

}

}

}

if (mode == 0)

{

cout << "\n";

}

}

void PrintArray(int\*\* array, int arrayBorder)

{

cout << " ";

for (int columnNumber = 0; columnNumber < arrayBorder; columnNumber++)

{

cout << setw(4) << columnNumber;

}

cout << "\n\n";

for (int row = 0; row < arrayBorder; row++)

{

cout << setw(2) << row << " ";

for (int column = 0; column < arrayBorder; column++)

{

cout << setw(4) << array[row][column];

}

cout << '\n';

}

cout << "\n";

}

int SumOfDigits(int number)

{

if (abs(number) < 10)

{

return abs(number);

}

else

{

return abs(number) % 10 + abs(number / 10);

}

}

int FindSideDiagonalSameRowNumber(int\*\* array, int lengthOfRow, int numberOfRow)

{

return array[numberOfRow][lengthOfRow - numberOfRow - 1];

}

void IncreaseArrayElements(int\*\* array, int arrayBorder)

{

for (int row = 0; row < arrayBorder; row++)

{

for (int column = 0; column < arrayBorder; column++)

{

if (column != arrayBorder - row - 1)

{

array[row][column] += SumOfDigits(FindSideDiagonalSameRowNumber(array, arrayBorder, row));

}

}

array[row][arrayBorder - row - 1] += SumOfDigits(array[row][arrayBorder - row - 1]);

}

}

void CheckSignAlternation(int\*\* array, int arrayBorder)

{

int row{};

cout << "Введите номер строки: ";

cin >> row;

if (row < 0 || row >= arrayBorder)

{

cout << "Ввод несуществующей строки\n";

return;

}

bool plusIsFound = false;

bool minusIsFound = false;

for (int column = 0; column < arrayBorder; column++)

{

if (array[row][column] > 0)

{

plusIsFound = true;

}

else if(array[row][column] < 0)

{

minusIsFound = true;

}

}

if (plusIsFound && minusIsFound)

{

cout << "Произошло чередование знаков\n";

}

else

{

cout << "Чередование знаков не произошло\n";

}

cout << "\n";

}

void FindZeroElement(int\*\* array, int arrayBorder)

{

int foundRow = -1;

int foundColumn = -1;

int firstRowOfSearch = arrayBorder / 3;

cout << "Первая строчка поиска: " << firstRowOfSearch << '\n';

for (int row = firstRowOfSearch; row < arrayBorder; row++)

{

for (int column = 0; column < arrayBorder; column++)

{

if (array[row][column] == 0)

{

foundRow = row;

foundColumn = column;

break;

}

}

if (foundRow != -1)

{

break;

}

}

if (foundRow != -1)

{

cout << "Первый нулевой элемент находится в " << foundRow << " строке " << foundColumn << " столбце\n";

}

else

{

cout << "Нулевых элементов в области поиска нет\n";

}

cout << "\n";

}

**Таблица идентификаторов:**

|  |  |  |
| --- | --- | --- |
| **Идентификаторы** – названия, даваемые экземплярам типов (имена переменных, функций, пространств и т.д.)  Нужно использовать американский английский, нотация выбирается в зависимости от вида идентификатора, название должно быть легко читаемым и понятным, допускается использование общепринятых акронимов | | |
| Имя идентификатора | Нотация | Условный перевод |
| MenuCodes | Upper Camel Case | Коды команд меню |
| PRINT\_ARRAY | Upper Case Snake Case | Код команды вывода массива |
| TASK\_1 | Upper Case Snake Case | Код команды решения первого задания |
| TASK\_2 | Upper Case Snake Case | Код команды решения второго задания |
| TASK\_3 | Upper Case Snake Case | Код команды решения третьего задания |
| EXIT | Upper Case Snake Case | Код команды выхода |
| MINIMUM\_RANDOM\_NUMBER | Upper Case Snake Case | Минимальное случайное число |
| MAXIMUM\_RANDOM\_NUMBER | Upper Case Snake Case | Максимальное случайное число |
| MINIMUM\_ARRAY\_BORDER | Upper Case Snake Case | Наименьший размер матрицы |
| MAXIMUM\_ARRAY\_BORDER | Upper Case Snake Case | Наибольший размер матрицы |
| Menu | Upper Camel Case | Меню |
| FillArray | Upper Camel Case | Заполнить массив |
| PrintArray | Upper Camel Case | Вывести массив |
| SumOfDigits | Upper Camel Case | Сложить цифры числа |
| FindSideDiagonalSameRowNumber | Upper Camel Case | Найти значение числа, находящегося на побочной диагонали той же строки |
| IncreaseArrayElements | Upper Camel Case | Увеличить значения элементов массива |
| CheckSignAlternation | Upper Camel Case | Проверить чередование знаков |
| FindZeroElement | Upper Camel Case | Найти нулевой элемент |
| arrayBorder | Lower Camel Case | Размер матрицы |
| mode | Lower Camel Case | Режим заполнения массива |
| array | Lower Camel Case | Массив |
| code | Lower Camel Case | Код команды меню |
| columnNumber | Lower Camel Case | Номер столбца |
| row | Lower Camel Case | Номер строки |
| column | Lower Camel Case | Номер столбца |
| number | Lower Camel Case | Число |
| lenghtOfRow | Lower Camel Case | Длина строки матрицы |
| numberOfRow | Lower Camel Case | Номер строки |
| plusIsFound | Lower Camel Case | Найдено положительное число |
| minusIsFound | Lower Camel Case | Найдено отрицательное число |
| foundRow | Lower Camel Case | Найденный номер строки |
| foundColumn | Lower Camel Case | Найденный номер столбца |
| firstRowOfSearch | Lower Camel Case | Номер строки, с которой начинается поиск |
| В идентификаторах недопустимо: | | |
| 1. Использование сокращений | | |
| 1. Транслит | | |
| 1. Венгерская нотация | | |
| В идентификаторах следует избегать: | | |
| 1. Сленг | | |
| 1. Использование ранее использованного названия, отличающегося только регистром | | |
| 1. Использование ключевых слов | | |
| 1. Просторечие | | |
| **Акронимы** – вид аббревиатуры, сокращение, которое можно произнести слитно, а не по буквам  Акронимы, состоящие из 3 и более символов, подчиняются рекомендациям обычных слов, из 2 символов – двумя строчными (Lower Camel Case) или двумя заглавными (Upper Camel Case) символами, кроме Id и Ok. Допускается использование общепринятых акронимов или акронимов, распространенных в данной предметной области | | |
| Акронимы | Нотация | Дешифровка |
|  |  |  |
| В акронимах недопустимо: | | |
| 1. Использование не очевидных сокращений | | |
| В акронимах следует избегать: | | |
| 1. Использование не общепринятых акронимов | | |
| **Именование типов**  Должен использоваться стиль Upper Camel Case, название должно быть существительным или именной группой, интерфейсы снабжаются префиксом I, абстрактные классы – префиксом Abstract или суффиксом Base, классы-исключения – суффиксом Exception, атрибуты – суффиксом Attribute. Идентификатор потомка должен указывать на различие и сходство с предком | | |
| Типы | Нотация | Дешифровка |
| MenuCodes | Upper Camel Case | Коды команд меню |
| При объявлении типов недопустимо: | | |
| 1. Прибавление префиксов или суффиксов, указывающих на то, что идентификатор является именем класса или структуры | | |
| 1. Использование одного и того же имени в идентификаторе класса и пространства имен | | |
| 1. Использование суффиксов в имени перечисления, указывающих на перечисление | | |
| **Процедуры и функции** – участки кода, которые можно вызывать множество раз в других местах кода  Должен использоваться стиль Upper Camel Case, имена должны быть глаголами или глагольными фразами. Следует выработать единый согласованный лексикон и следовать ему. | | |
| Также следует использовать общепринятые антонимы.  В прототипе функции необходимо указать тип возвращаемого элемента, идентификатор, а в круглых скобках – параметры, принимаемые функцией или процедурой. Тело функции указывается в фигурных скобках. | | |
| Процедура или функция | Нотация | Дешифровка |
| Menu | Upper Camel Case | Меню |
| FillArray | Upper Camel Case | Заполнить массив |
| PrintArray | Upper Camel Case | Вывести массив |
| SumOfDigits | Upper Camel Case | Сложить цифры числа |
| FindSideDiagonalSameRowNumber | Upper Camel Case | Найти значение числа, находящегося на побочной диагонали той же строки |
| IncreaseArrayElements | Upper Camel Case | Увеличить значения элементов массива |
| CheckSignAlternation | Upper Camel Case | Проверить чередование знаков |
| FindZeroElement | Upper Camel Case | Найти нулевой элемент |
| При именовании недопустимо: | | |
| 1. Использование имен, не являющихся глаголами | | |
| При именовании следует избегать: | | |
| 1. Название объекта, над которым выполняется действие, если это ясно из контекста | | |
| 1. Имена типов | | |
| **Константы** – переменные, которые можно лишь объявить и инициализировать, в дальнейшем поменять их значение будет невозможно  Должен использоваться Upper Case Snake Case, использование не общепринятых констант должно быть сведено к минимуму, вместо них следует использовать функции, возвращающие значение определенного числа. «Магические числа» следует заменять константами, кроме чисел 0 и 1. Вместо числовых констант следует использовать перечисления. | | |
| Константы | Нотация | Дешифровка |
| PRINT\_ARRAY | Upper Case Snake Case | Код команды вывода массива |
| TASK\_1 | Upper Case Snake Case | Код команды решения первого задания |
| TASK\_2 | Upper Case Snake Case | Код команды решения второго задания |
| TASK\_3 | Upper Case Snake Case | Код команды решения третьего задания |
| EXIT | Upper Case Snake Case | Код команды выхода |
| MINIMUM\_RANDOM\_NUMBER | Upper Case Snake Case | Минимальное случайное число |
| MAXIMUM\_RANDOM\_NUMBER | Upper Case Snake Case | Максимальное случайное число |
| MINIMUM\_ARRAY\_BORDER | Upper Case Snake Case | Наименьший размер матрицы |
| MAXIMUM\_ARRAY\_BORDER | Upper Case Snake Case | Наибольший размер матрицы |
| **Локальные переменные** – именованная ячейка памяти, имеющая локальную область видимости  Должен использоваться Lower Camel Case, переменные класса должны иметь префикс m\_ | | |
| Локальные переменные | Нотация | Дешифровка |
| arrayBorder | Lower Camel Case | Размер матрицы |
| mode | Lower Camel Case | Режим заполнения массива |
| array | Lower Camel Case | Массив |
| code | Lower Camel Case | Код команды меню |
| columnNumber | Lower Camel Case | Номер столбца |
| row | Lower Camel Case | Номер строки |
| column | Lower Camel Case | Номер столбца |
| number | Lower Camel Case | Число |
| lenghtOfRow | Lower Camel Case | Длина строки матрицы |
| numberOfRow | Lower Camel Case | Номер строки |
| plusIsFound | Lower Camel Case | Найдено положительное число |
| minusIsFound | Lower Camel Case | Найдено отрицательное число |
| foundRow | Lower Camel Case | Найденный номер строки |
| foundColumn | Lower Camel Case | Найденный номер столбца |
| firstRowOfSearch | Lower Camel Case | Номер строки, с которой начинается поиск |

**Вывод:** в ходе работы были получены практические навыки работы в команде, использования стандарта кодирования, составления таблицы идентификаторов.