САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ

УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Практическая работа №7

«Применение структур»

Выполнил: Мороз И.О.

Группа К3120

Проверил: Осипов Н.А.

Санкт-Петербург

2018 г.

# Упражнение 1.

1. Для демонстрации работы структур был создан новый проект
2. Была объявлена структура Distance

struct Distance

{

int feet;

double inches;

};

1. Для работы с структурой были созданы методы для ввода, сложения и вывода структуры в консоль

Distance AddDist(Distance d1, Distance d2)

{

Distance d;

d.feet = d1.feet + d2.feet;

d.inches = d1.inches + d2.inches;

if (d.inches >= 12.0)

{

d.feet++;

d.inches -= 12.0;

}

return d;

}

Distance InputDist()

{

Distance d;

cout << "\nВведите число футов: ";

cin >> d.feet;

cout << "\nВведите число дюймов: ";

cin >> d.inches;

return d;

}

void ShowDist(Distance d)

{

cout << d.feet << "\'-" << d.inches << "\"\n";

}

1. Внутри метода main были объявлены 3 переменные структуры Distance. Значения первой вводятся с клавиатуры, значения второй присваиваются при объявлении, значения третьей получены при помощи суммирования значений первой и второй переменных

setlocale(LC\_ALL, "Russian");

Distance d1 = InputDist();

Distance d2 = { 1, 6.25 };

Distance d3 = AddDist(d1, d2);

ShowDist(d1);

ShowDist(d2);

ShowDist(d3);

system("pause");

return 0;

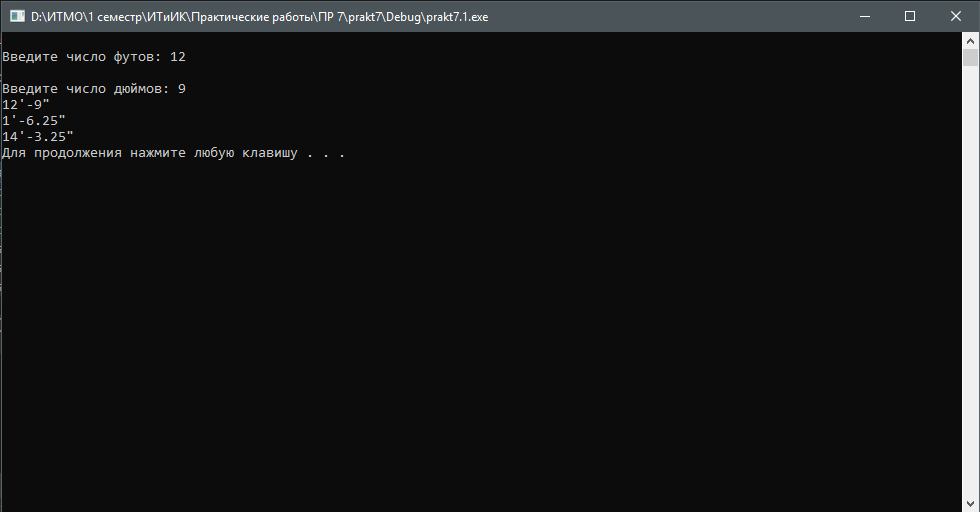


Рисунок 1 – результат работы программы

1. Определение функции ShowDist() было перенесено в структуру. Теперь вывод значения переменной структуры выглядит так

d1.ShowDist();

d2.ShowDist();

d3.ShowDist();

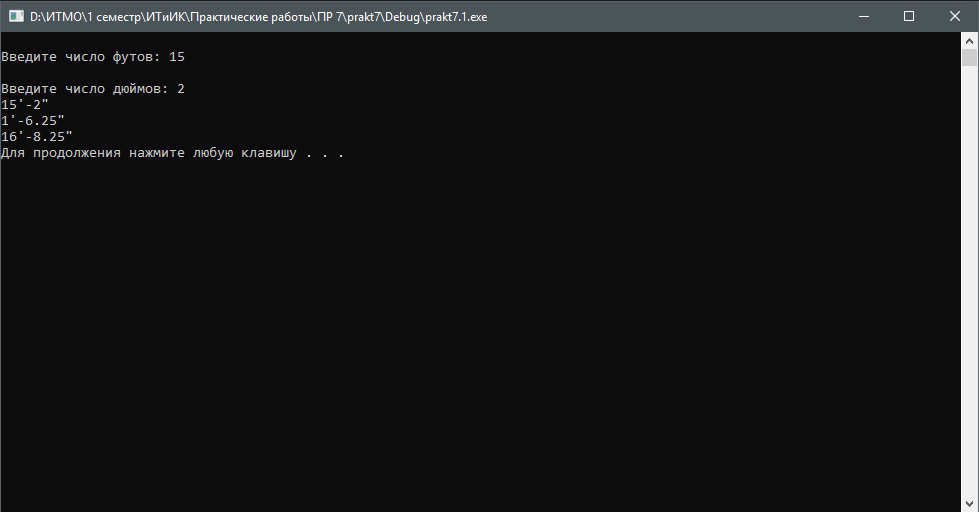


Рисунок 2 – изменения не повлияли на работу программы

# Упражнение 2.

1. Были изменены типы передаваемых в методы переменных. Теперь в методы передаётся ссылка на переменную, а не сама переменная. Для защиты значения переменной она была объявлена как const

Distance AddDist(const Distance &d1, const Distance &d2)

{

Distance d;

d.feet = d1.feet + d2.feet;

d.inches = d1.inches + d2.inches;

if (d.inches >= 12.0)

{

d.feet++;

d.inches -= 12.0;

}

return d;

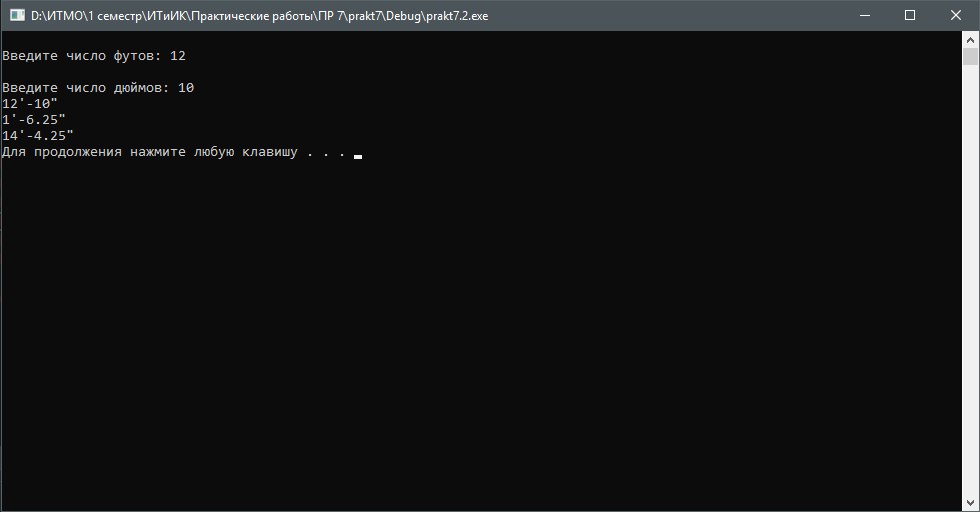
}

void ShowDist(const Distance &d)

{

cout << d.feet << "\'-" << d.inches << "\"\n";

}

Рисунок 3 – после передачи ссылок в методы их результат не изменился

# Упражнение 3.

1. При помощи цикла for был реализован ввод и вывод динамического массива masDist структуры Distance в консоль

int n;

cout << "Введите размер массива расстояний ";

cin >> n;

Distance \* masDist = new Distance[n];

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

{

masDist[i] = InputDist();

}

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

{

ShowDist(masDist[i]);

}

delete[] masDist;

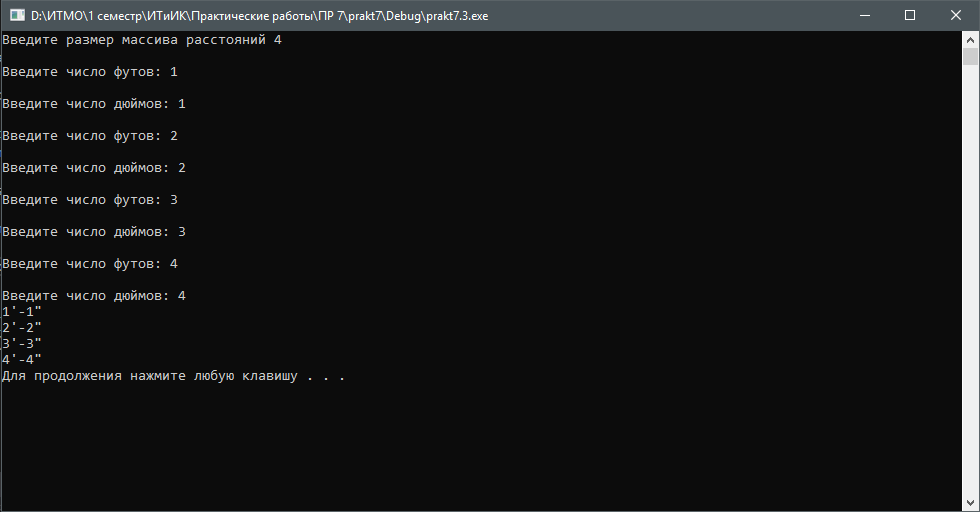


Рисунок 4 – пример работы программы

1. В коде программы была реализована возможность подсчёта суммы длин всех переменных массива при помощи цикла for

Distance dSum = { 0, 0.0 };

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

{

dSum = AddDist(dSum, masDist[i]);

}

cout << "Сумма длин всех элементов массива: ";

ShowDist(dSum);

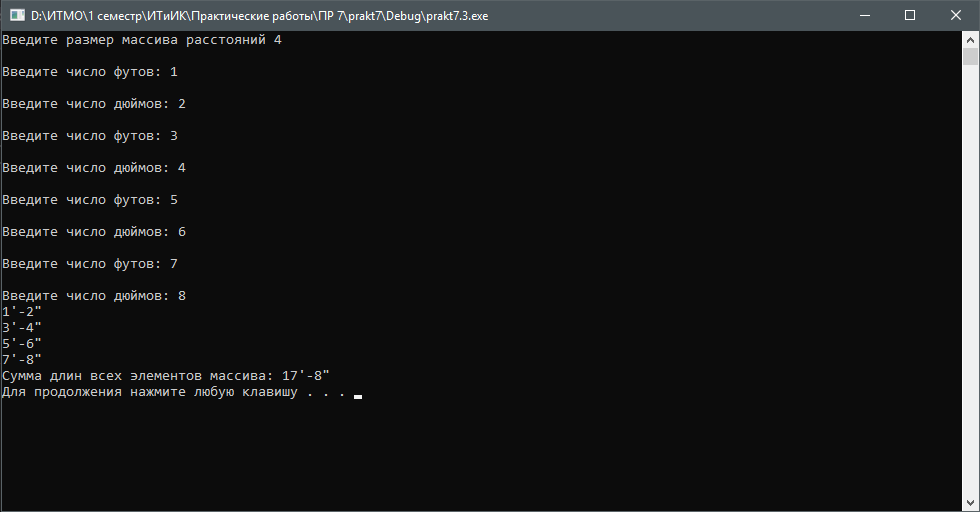


Рисунок 5 – вывод суммы элементов в консоль

# Вывод.

В практической работе были рассмотрены структуры, а также принципы работы с нимим. Были написаны алгоритмы ввода, вывода, сложения переменных структур. Был рассмотрен вариант создания массива структур, а также для закрепления материала был написан алгоритм нахождения суммы всех элементов массива.

# Контрольные вопросы

1. Реализовать передачу суммы без возврата этого значения через return.

void AddDist(const Distance &d1, const Distance &d2, Distance &d3)

{

d3.feet = d1.feet + d2.feet;

d3.inches = d1.inches + d2.inches;

if (d3.inches >= 12.0)

{

d3.feet++;

d3.inches -= 12.0;

}

}

1. В чём отличия между структурами и классами?

По умолчанию доступ к структуре – public, а к классу – private.