**Глава 2**
**Исходный код**

```
#include <iostream>
using namespace std;
inline int vInput(int* ins);
inline int vMulti(int* ins);
inline int vSum(int* ins);
inline int vSub(int* ins);
inline int vEq(int* ins);

int main()
{
    int n = 73;
    int m;
    int x[2], y[2];
    x[0] = 0;
    x[1] = 0;
    puts("WARNING: X been set to [0,0] as default");
    while (n != 0) {
        puts("\n Choose action: \n 0) Exit \n 1) Input/reset \n 2) summation
+ \n 3) subtrction - \n 4) scalar * \n 5) equalence =");
        try {
            if (scanf_s("%d", &n) != 1) {
                throw 0;
            }
            switch (n) {
            case 0:
                puts("exit...");
                break;
            case 1:
                vInput(&x[0]);
                break;
            case 2:
                vSum(&x[0]);
                break;
            case 3:
                vSub(&x[0]);
                break;
            case 4:
                vMulti(&x[0]);
                break;
            case 5:
                vEq(&x[0]);
```

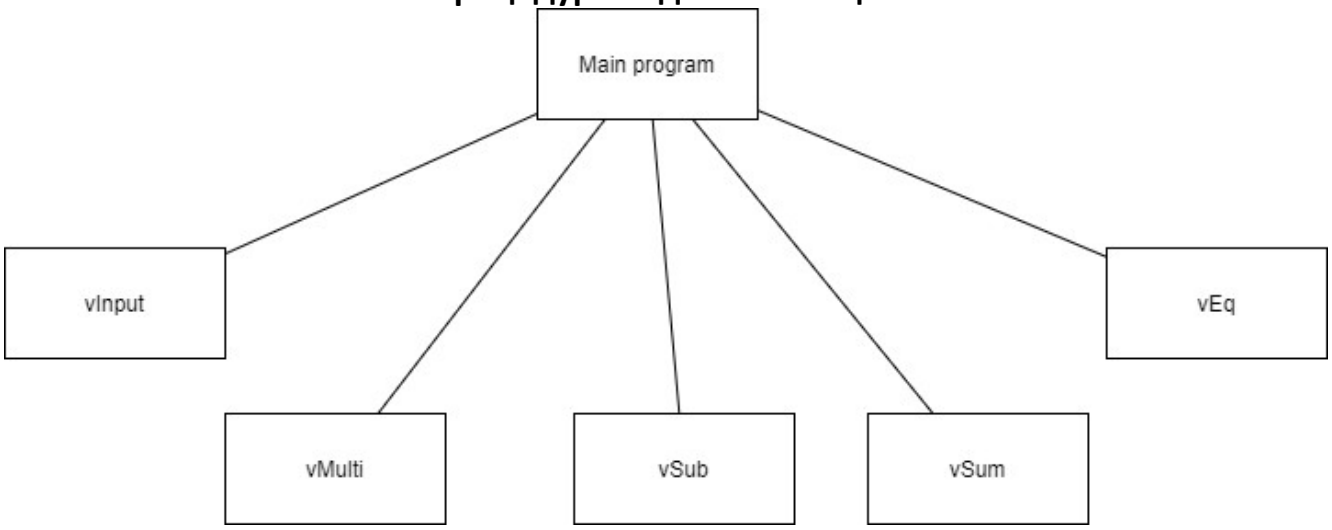
```
                        break;
                default:
                        puts("Unknown operation id, please retry \n");
                }
        }
        catch (int excep) {
                printf("Anomaly input, please retry (code %d) \n Error
finishing... \n", excep);
                n = 0;
        }
        printf("Current X = [%d, %d] \n", x[0], x[1]);
    }
}

inline int vInput(int* ins) {
    int a, b;
    puts("Enter coords in formar %d %d");
    if (scanf_s("%d %d", &a, &b) != 2) {
        throw 1;
    }
    *ins = a;
    *(ins + 1) = b;
    return 0;
}

inline int vMulti(int* ins) {
    int a;
    puts("Enter num to multiply");
    if (scanf_s("%d", &a) != 1) {
        throw 1;
    }
    *ins = a * (*ins);
    *(ins + 1) = a * (*(ins + 1));
    return 0;
}

inline int vSum(int* ins) {
    int a, b;
    puts("Enter coords in formar %d %d");
    if (scanf_s("%d %d", &a, &b) != 2) {
        throw 1;
    }
    *ins = a + (*ins);
    *(ins + 1) = b + (*(ins + 1));
```

```
    return 0;
}

inline int vSub(int* ins) {
    int a, b;
    puts("Enter coords in formar %d %d");
    if (scanf_s("%d %d", &a, &b) != 2) {
            throw 1;
    }
    *ins = -a + (*ins);
    *(ins + 1) = -b + (*(ins + 1));
    return 0;
}

inline int vEq(int* ins) {
    int a, b;
    puts("Enter coords in formar %d %d");
    if (scanf_s("%d %d", &a, &b) != 2) {
            throw 1;
    }
    if ((*ins == a) && (*(ins + 1) == b)) {
            puts("Vectors equal \n");
    }
    else {
            puts("Vectors dont equal \n");
    }
    return 0;
}
```

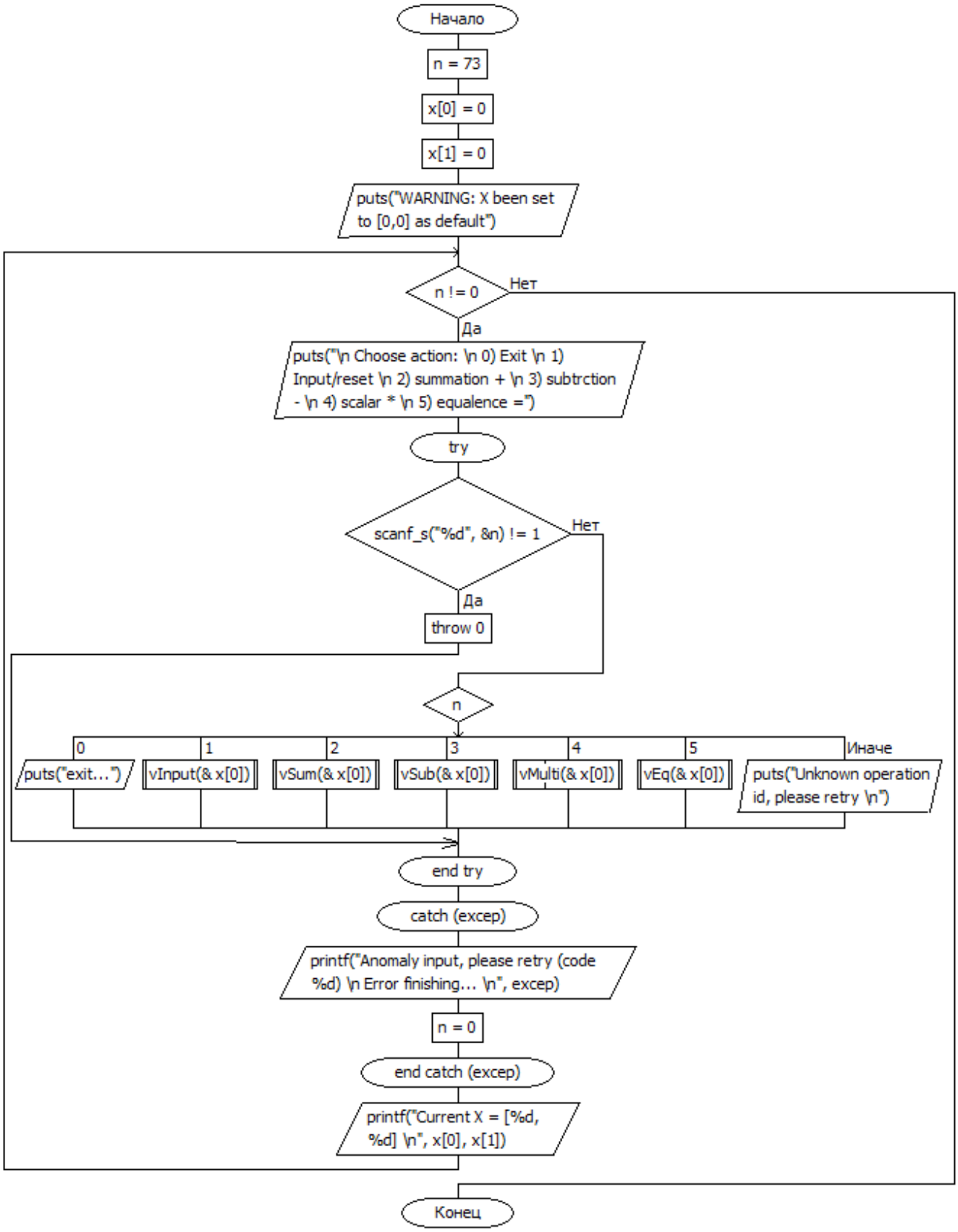
**Тесты**

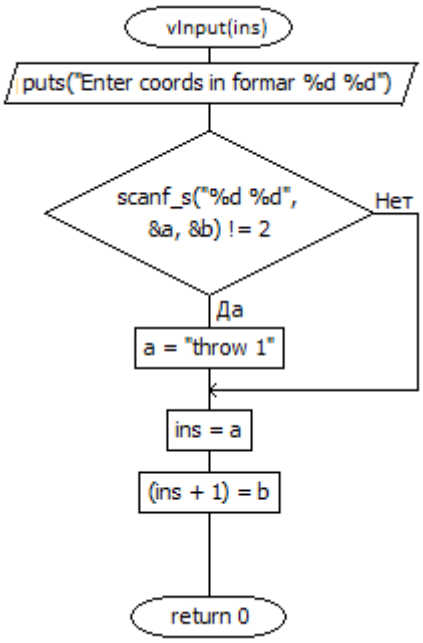
| Входные данные | Ожидаемые выходные данные | Выходные данные |
|---|---|---|
| 1<br>2 4 | Current X = [2, 4] | Current X = [2, 4] |
| 2<br>1 1 | Current X = [3, 5] | Current X = [3, 5] |
| 4<br>2 | Current X = [6, 10] | Current X = [6, 10] |

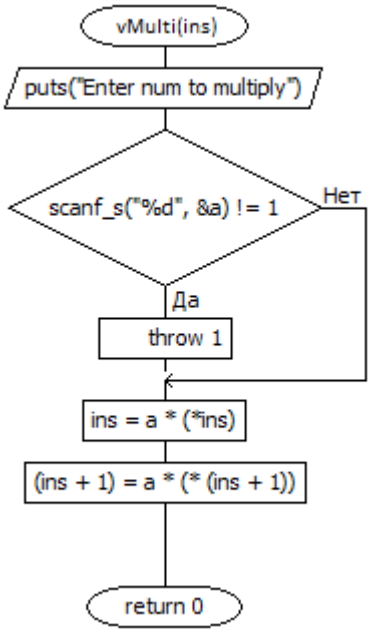
## Процедурная декомпозиция

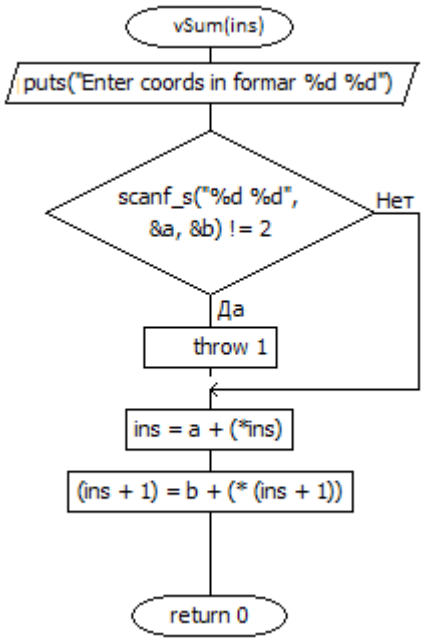


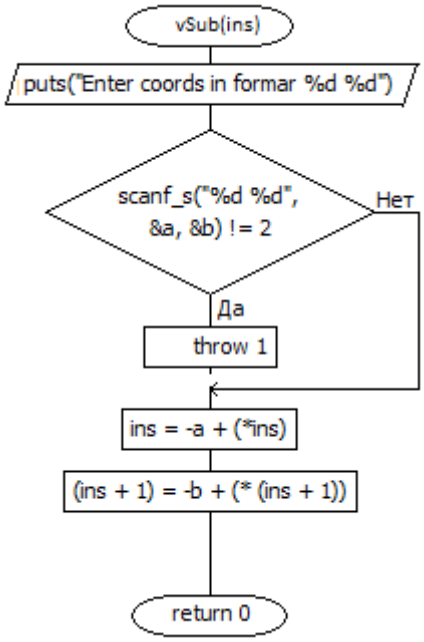
## Схема алгоритма

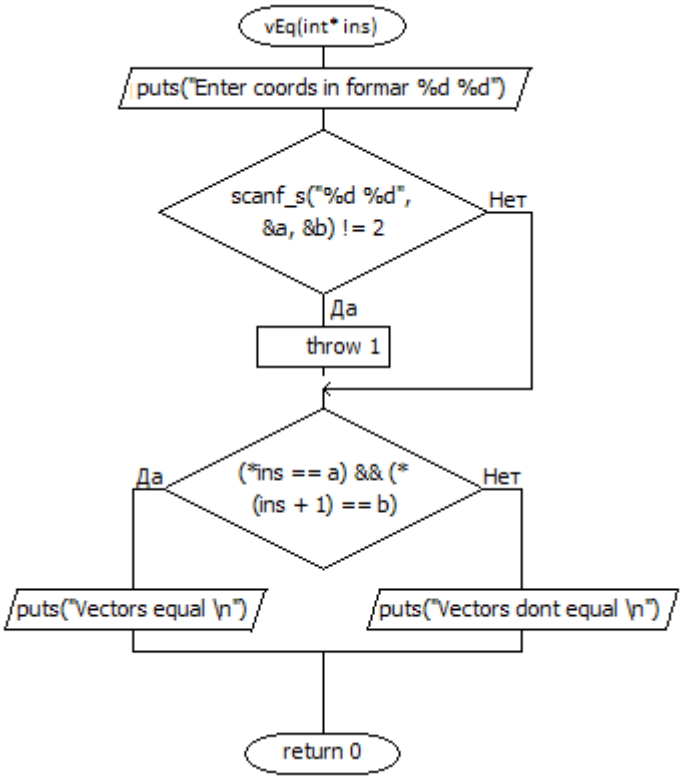



vInput(ins)
puts("Enter coords in formar %d %d")
scanf_s("%d %d", &a, &b) != 2
Нет
Да
a = "throw 1"
ins = a
(ins + 1) = b
return 0


vMulti(ins)
puts("Enter num to multiply")
scanf_s("%d", &a) != 1
Нет
Да
throw 1
ins = a * (*ins)
(ins + 1) = a * (* (ins + 1))
return 0


vSum(ins)
puts("Enter coords in formar %d %d")
scanf_s("%d %d", &a, &b) != 2
Нет
Да
throw 1
ins = a + (*ins)
(ins + 1) = b + (* (ins + 1))
return 0


vSub(ins)
puts("Enter coords in formar %d %d")
scanf_s("%d %d", &a, &b) != 2
Нет
Да
throw 1
ins = -a + (*ins)
(ins + 1) = -b + (* (ins + 1))
return 0


vEq(int* ins)
puts("Enter coords in formar %d %d")
scanf_s("%d %d", &a, &b) != 2
Нет
Да
throw 1
(*ins == a) && (* (ins + 1) == b)
Да
Нет
puts("Vectors equal \n")
puts("Vectors dont equal \n")
return 0

## Вывод

- функции C++ не имеют значительных отличий от функций Delphi, однако поддерживают некоторые дополнительные опции, облегчающие процесс разработки и позволяющие оптимизировать программу
- Стоит отметить, что в отличие от Delphi в C++ нет аналога служебного слова var, поэтому для написания функций-процедур необходимо передавать указатели содержащие адрес изменяемой переменной