Основы программирования

ФИСТ 1 курс
Власенко
Олег
Федосович

Лекция 7

Стандартные типы. Трассировка.

Основные типы данных

Тип	Длина байт	Диапазон значений	Минимальное значение	Максимальное значение
char				
double				
short				
long				

Составить таблицу символов

```
#include <stdio.h>

void main() {
	char ch = ' ';

	int i = 0;
	do {
		printf("%4d--> '%c'\t", ch, ch);
		ch = ch + 1;
		i = i + 1;
	} while (i <= 256);
}
```

Основные типы данных (ASCII)

Тип	Длина байт	Диапазон значений	Минимальное значение	Максимальное значение
char	**1**	**256**	**-128**	**+127**
double				
short				
long				

Подсчитать MAX short

```
void main() {
        short i = 1;
        long  n = 0;
        do {
                i = i + 1;
                n = n + 1;
        } while (i > 0);
        printf("%li\n", n);
}
```

Основные типы данных

Тип	Длина байт	Диапазон значений	Минимальное значение	Максимальное значение
char	1	256	-128	+127
double				
short				**32767**
long				

Сколько байт в short и long?

```
void main() {

        short i;
        long l;

        printf("sizeof short = %d\n", sizeof(i));
        printf("sizeof long = %d\n", sizeof(l));
}
```

Основные типы данных

Тип	Длина байт	Диапазон значений	Минимальное значение	Максимальное значение
char	1	256	-128	+127
double				
short	**2**			32767
long	**4**			

Основные типы данных

Тип	Длина байт	Диапазон значений	Минимальное значение	Максимальное значение
char	1	2^8 = 256	-128	+127
double	**8**	**IEEE 754 standard**	**2.22507e-308**	**1.79769e+308**
short	2	**2^16= 65 536**	**-32 768**	32767
long	4	**2^32 = +4,294,967,296**	**–2,147,483,648**	**+2,147,483,647**

Строка форматирования

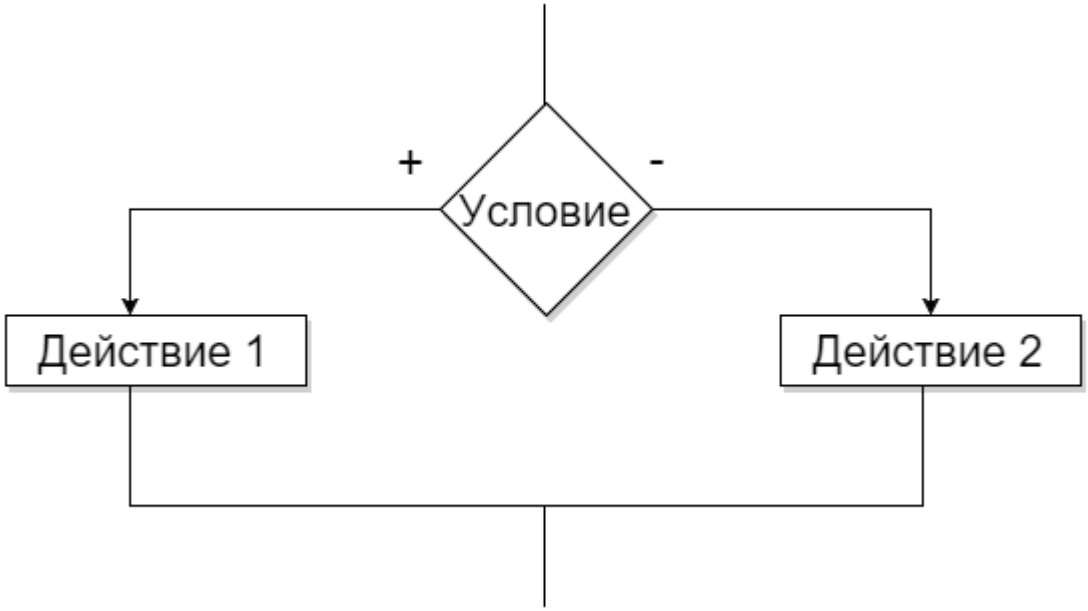
Тип	scanf/printf
char	%c
short	%hi
int	%d или %i
long	%li
float	%f
double	%lf
long double	%Lf

Консоль – что из себя представляет.
Знакоместо – что это такое.

Поиск корней квадратного уравнения

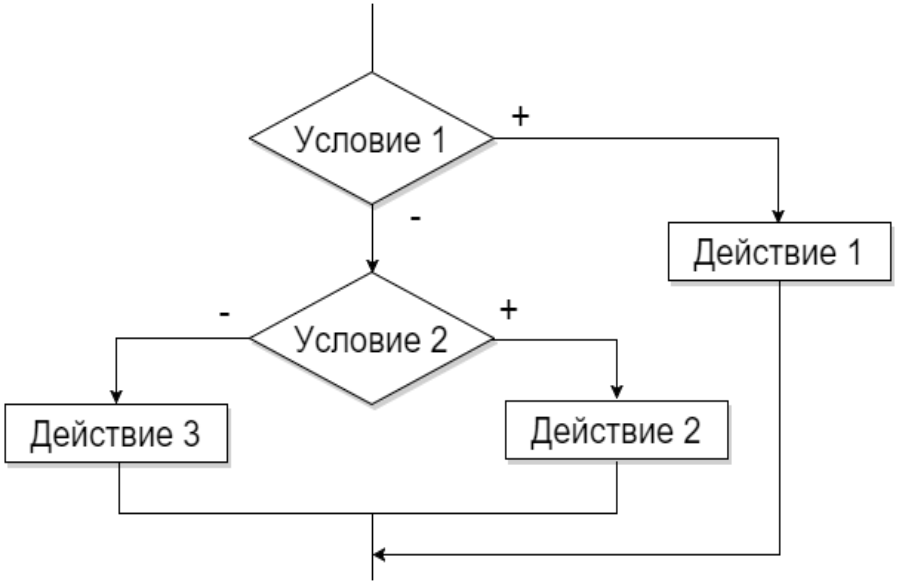
```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <math.h>

void main() {
		double a, b, c;
		double D;
		double x1, x2;

		scanf("%lf", &a);
		scanf("%lf", &b);
		scanf("%lf", &c);

		D = b * b - 4 * a * c;

		x1 = (-b + sqrt(D)) / (2 * a);
		x2 = (-b - sqrt(D)) / (2 * a);

		printf("x1 = %lf", x1);
		printf("x2 = %lf", x2);
}
```

Развилка




```
if (Условие)
        Действие1;
else
        Действие2;
```

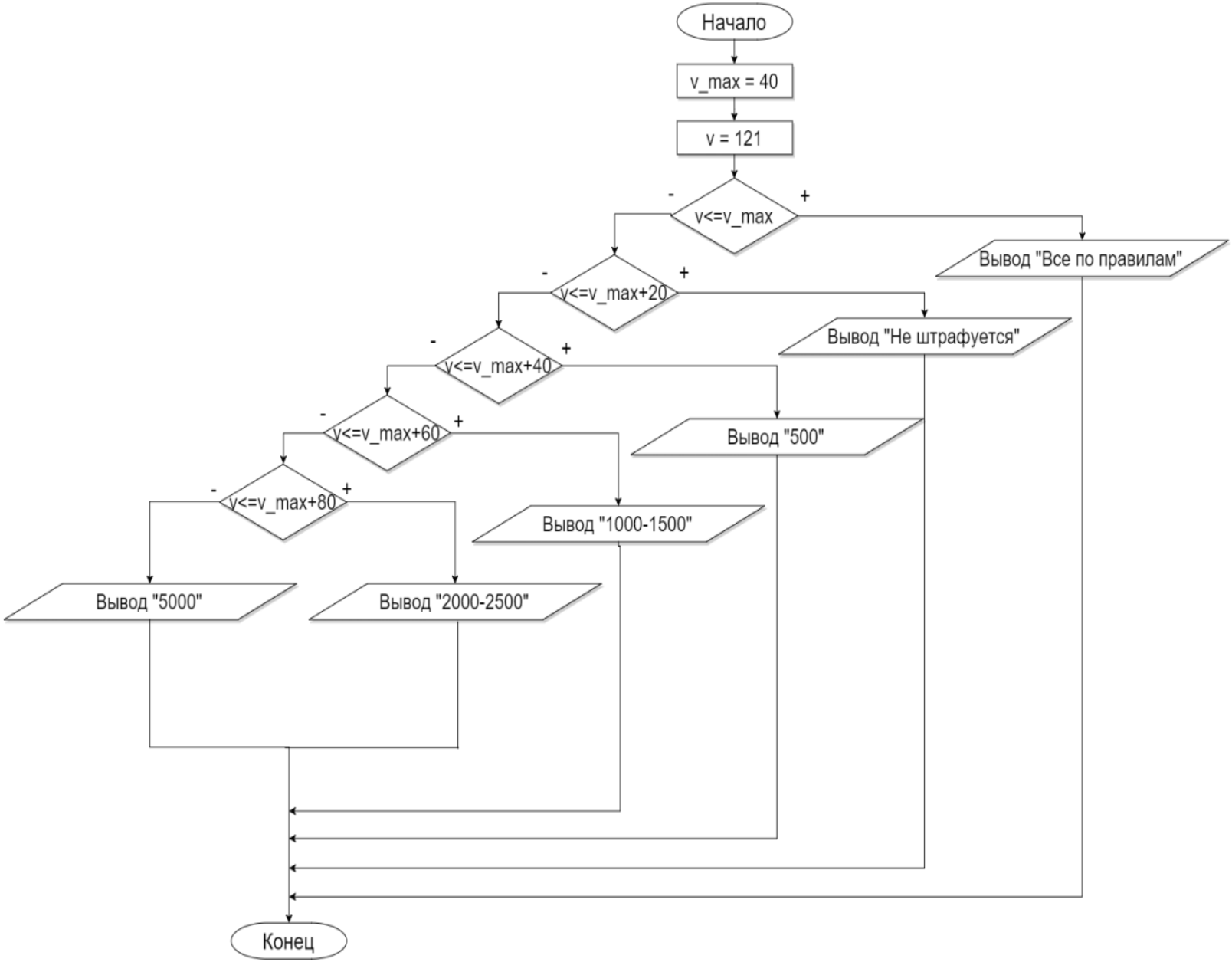
Вложенные развилки




```
if (Условие 1) {
        Действие 1
} else {
        if (Условие 2) {
                Действие 2
        } else {
                Действие 3
        }
}
```

```
if (Условие 1) {
        Действие 1
} else if (Условие 2) {
        Действие 2
} else {
        Действие 3
}
```

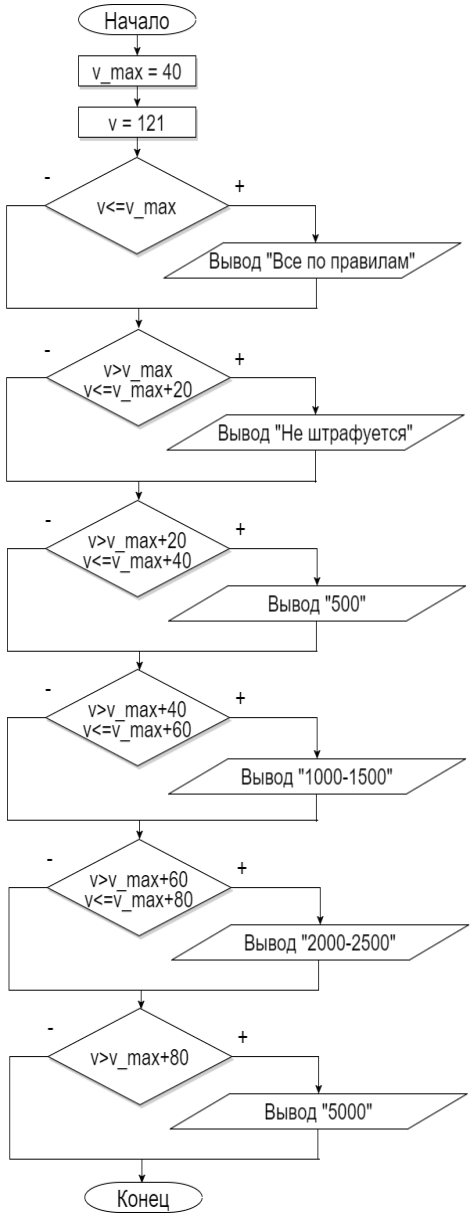
Штраф за превышение скорости



Штраф за превышение скорости – полная развилка

```
void main() {
        int v_max = 40;
        int v = 30;

        if (v <= v_max) {
                printf("All right!");
        } else if (v <= v_max + 20) {
                printf("No $$$");
        } else if (v <= v_max + 40) {
                printf("500");
        } else if (v <= v_max + 60) {
                printf("1000-1500");
        } else if (v <= v_max + 80) {
                printf("2000-2500");
        } else {
                printf("5000");
        }
}
```

Штраф за превышение скорости



Штраф за превышение скорости – усеченная развилка

```
void main() {
  int v_max = 40;
  int v = 70;

  if (v <= v_max) {
    printf("Все по правилам!");
  }
  if ((v > v_max) && (v <= v_max + 20)) {
   printf("не штрафуется");
  }
  if ((v > v_max + 20) && (v <= v_max + 40)) {
    printf("500");
  }
  if ((v > v_max + 40) && (v <= v_max + 60)) {
     printf("1000-1500");
  }
  if ((v > v_max + 60) && (v <= v_max + 80)) {
     printf("2000-2500");
  }
  if (v > v_max + 80) {
      printf("5000");
  }
}
```

Логические операции

Оператор	Описание
&&	Логическое И (AND)
||	Логическое ИЛИ (OR)
!	Логическое унарное НЕ (NOT)

A	!A
0	1
1	0

A	B	A && B	A || B
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

```
if (time < 7.00 || day >= 6)  rest();

if (!closed && money > 1000)  eat();
```

Домашнее задание

1. Дойти до предела long. Найти такую задачу, где нужны целые числа, а возможностей long недостаточно. (В идеале – реализовать её в коде)
2. Дойти до предела double. Найти задачу, где возможностей double недостаточно для вычислений.

Трассировка циклов

Задача 1: Вывести в консоль числа от 1 до N

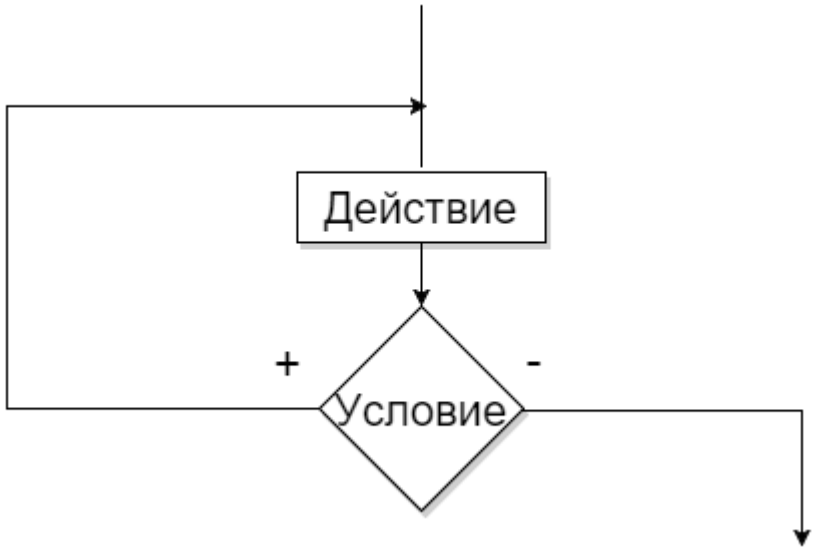
```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <math.h>

void main() {
	int i;
	int n;

	scanf("%d", &n);
	…









}
```

Задача 1: Вывести в консоль числа от 1 до N

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#include <math.h>

void main() {
	int i;
	int n;

	scanf("%d", &n);
	i = 1;
	do {
		printf("%d ", i);
		i++;
	} while (i <= n);

	printf("\n");
}
```

Задача 1.2: Блок-схема

```
do {
        Действие;
} while (Условие);
```



Задача 1.3: Трассировка

I	N	I <= N	Ввод	вывод

Задача 1.3: Трассировка

I	N	I <= N	Ввод	вывод
			4	

Задача 1.3: Трассировка

I	N	I <= N	Ввод	вывод
	4		4	

Задача 1.3: Трассировка

I	N	I <= N	Ввод	вывод
1	4		4	

Задача 1.3: Трассировка

I	N	I <= N	Ввод	вывод
1	4		4	1

Задача 1.3: Трассировка

I	N	I <= N	Ввод	вывод
1+1=2	4		4	1

Задача 1.3: Трассировка

I	N	I <= N	Ввод	вывод
1+1=2	4	2 <= 4 ДА	4	1

Задача 1.3: Трассировка

I	N	I <= N	Ввод	вывод
1+1=2	4	2 <= 4 ДА	4	1
				2

Задача 1.3: Трассировка

I	N	I <= N	Ввод	вывод
1+1=2	4	2 <= 4 ДА	4	1
2+1=3				2

Задача 1.3: Трассировка

I	N	I <= N	Ввод	вывод
1+1=2	4	2 <= 4 ДА	4	1
2+1=3		3 <= 4 ДА		2

Задача 1.3: Трассировка

I	N	I <= N	Ввод	вывод
1+1=2	4	2 <= 4 ДА	4	1
2+1=3		3 <= 4 ДА		2
				3

Задача 1.3: Трассировка

I	N	I <= N	Ввод	вывод
1+1=2	4	2 <= 4 ДА	4	1
2+1=3		3 <= 4 ДА		2
3+1=4				3

Задача 1.3: Трассировка

I	N	I <= N	Ввод	вывод
1+1=2	4	2 <= 4 ДА	4	1
2+1=3		3 <= 4 ДА		2
3+1=4		4 <= 4 ДА		3

Задача 1.3: Трассировка

I	N	I <= N	Ввод	вывод
1+1=2	4	2 <= 4 ДА	4	1
2+1=3		3 <= 4 ДА		2
3+1=4		4 <= 4 ДА		3
				4

Задача 1.3: Трассировка

I	N	I <= N	Ввод	вывод
1+1=2	4	2 <= 4 ДА	4	1
2+1=3		3 <= 4 ДА		2
3+1=4		4 <= 4 ДА		3
4+1=5				4

Задача 1.3: Трассировка

I	N	I <= N	Ввод	вывод
1+1=2	4	2 <= 4 ДА	4	1
2+1=3		3 <= 4 ДА		2
3+1=4		4 <= 4 ДА		3
4+1=5		5 <= 4 НЕТ		4

Задача 1.3: Трассировка

I	N	I <= N	Ввод	вывод
1+1=2	4	2 <= 4 ДА	4	1
2+1=3		3 <= 4 ДА		2
3+1=4		4 <= 4 ДА		3
4+1=5		5 <= 4 НЕТ		4
				\n

Задача 2: Вывести в консоль N первых нечетных чисел

```
void main() {
        int i;
        int a;
        int n;

        scanf("%d", &n);
        a = 1;
        i = 1;
        …




        printf("\n");
}
```

Задача 2: Вывести в консоль N первых нечетных чисел

```
void main() {
	int i;
	int a;
	int n;

	scanf("%d", &n);
	a = 1;
	i = 1;
	do {
		printf("%d ", a);
		a += 2;
		i++;
	} while (i <= n);
	printf("\n");
}
```

Задача 2.2: Блок схема

Задача 2.3: Трассировка

N	I	A	I <= N	Ввод	Вывод
				3	

Задача 3: Вывести числа в консоль: 10 8 6 … 0

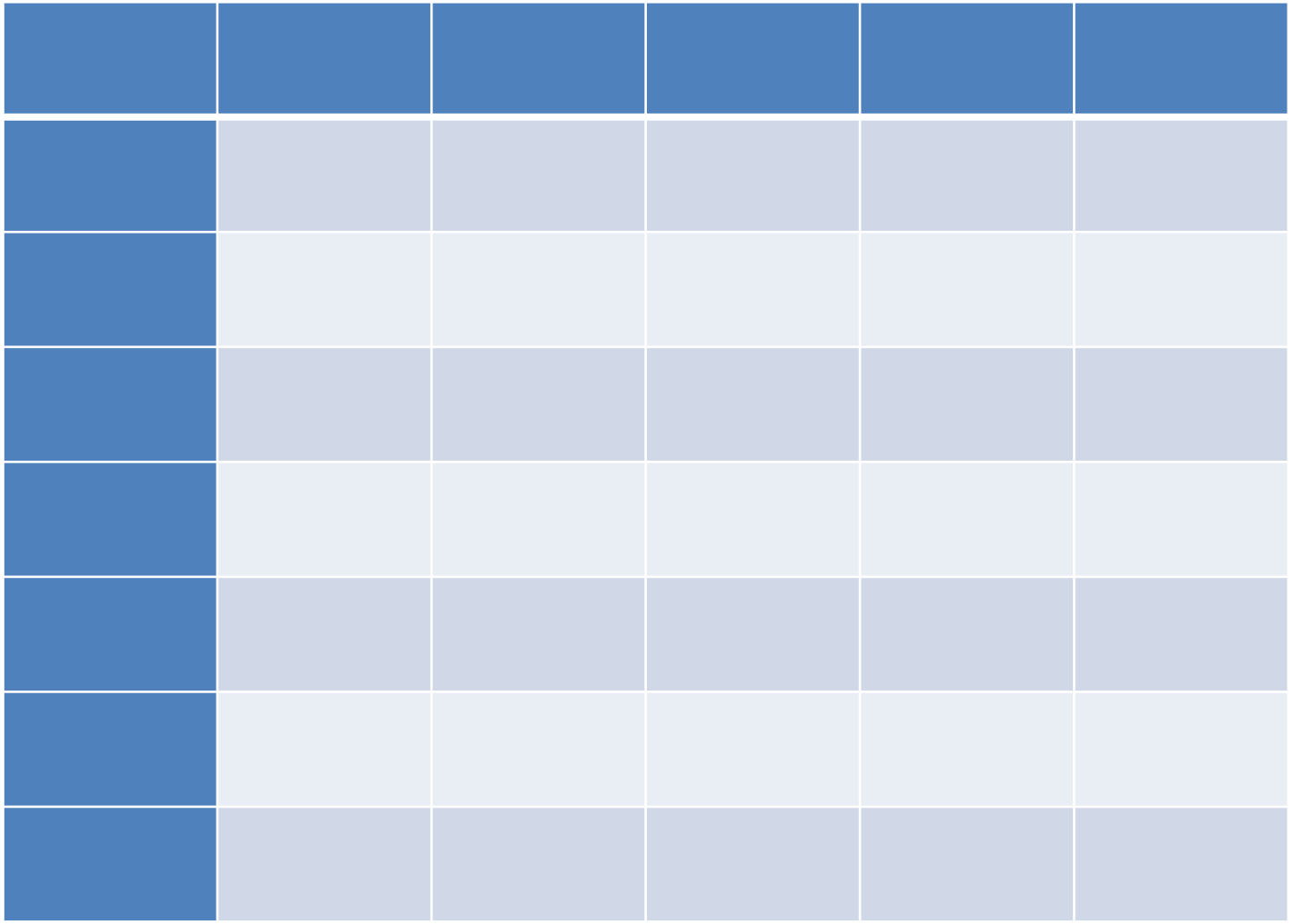
```
void main() {

        int i;


        i = 10;
        …


        printf("\n");

}
```

Задача 3: Вывести числа в консоль: 10 8 6 … 0

```
void main() {

	int i;


	i = 10;
	do {
		printf("%d ", i);
		i -= 2;
	} while (i >= 0);

	printf("\n");

}
```

Задача 3.2: Блок схема

Задача 3.3: Трассировка



Задача 3.3: Трассировка

I	I >= 0	Вывод			
10					

Задача 4: Вывести в консоль степени двойки

Пример ввода:

4

Пример вывода:

2^0 = 1

2^1 = 2

2^2 = 4

2^3 = 8

2^4 = 16

Задача 4: Вывести в консоль степени двойки

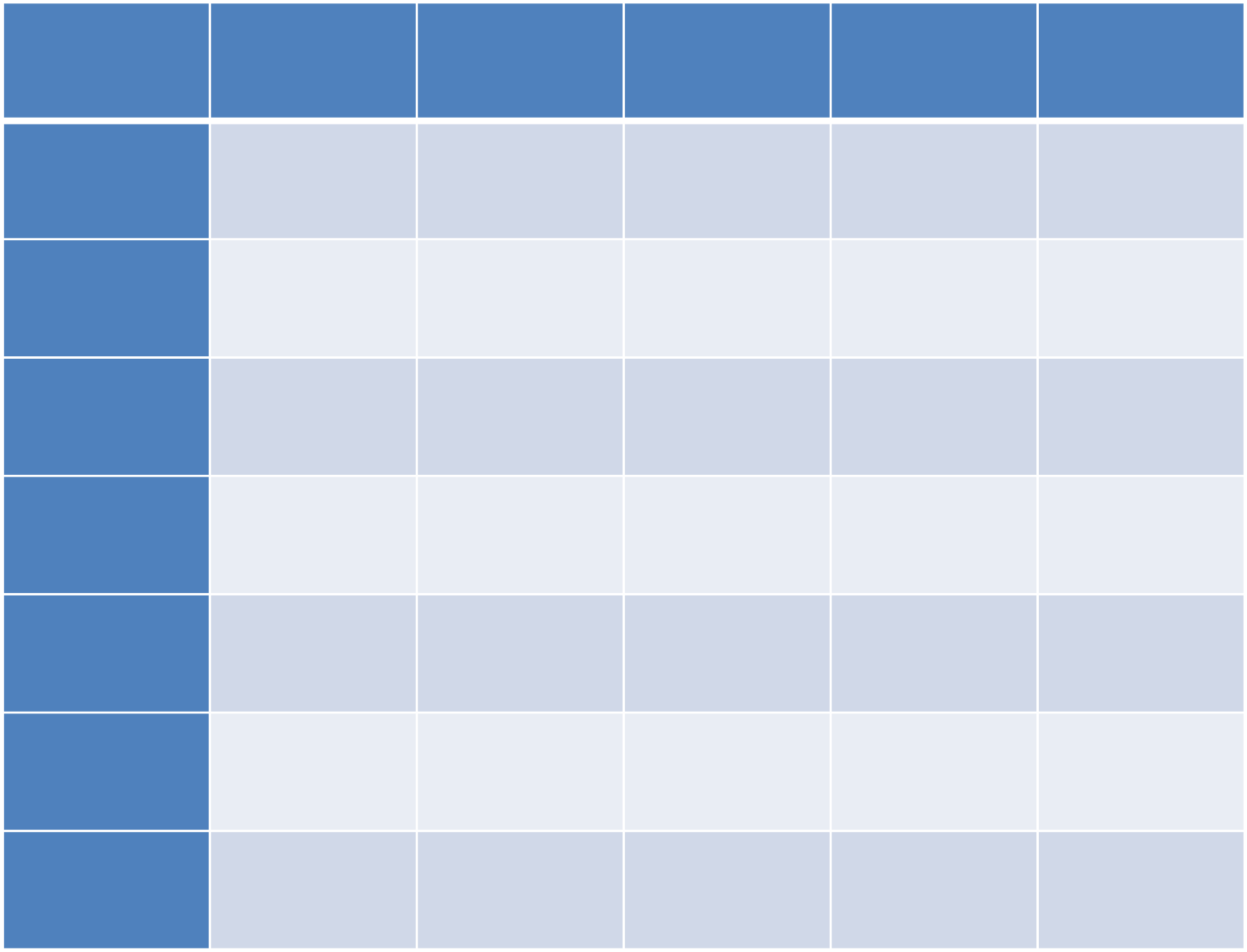
```
void main() {

        int i;
        int a;
        int n;

        scanf("%d", &n);

        …
                printf("2^%d = %d\n", i, a);
        …

}
```

Задача 4: Вывести в консоль степени двойки

```
void main() {

	int i;
	int a;
	int n;

	scanf("%d", &n);

	a = 1;
	i = 0;
	do {
		printf("2^%d = %d\n", i, a);
		a *= 2;
		i++;
	} while (i <= n);


}
```

Задача 4.2: Блок-схема

Задача 4.3: Трассировка



Задача 4.3: Трассировка

I	A	N	I <= N	Ввод	Вывод
		4			

Задача 5: Вычислить и вывести первые N чисел Фибоначчи

Числа Фибоначчи

0: 0 + 1 = 1

1: 1 + 1 = 2

2 : 1 + 2 = 3

3 : 2 + 3 = 5

4 : 3 + 5 = 8

5 : 5 + 8 = 13

6 : 8 + 13 = 21

7 : 13 + 21 = 34

8 : 21 + 34 = 55

9 : 34 + 55 = 89

... и т. д.

Задача 5: Вычислить и вывести первые N чисел Фибоначчи

```
void main() {
        int i;
        int a1,  a2, a3;
        int n;

        scanf("%d", &n);
        a1 = 0;
        a2 = 1;
        i = 1;
        do {
                printf("%d ", a2);
                a3 = a2 + a1;
                a1 = a2;
                a2 = a3;
                i++;
        } while (i <= n);
        printf("\n");
}
```

Задача 5.2: Блок-схема

Задача 5.3: Трассировка

Задача 5.3: Трассировка

I	N	A1	A2	A3	I <= N	Ввод	Вывод
	4						

Домашнее задание

1. Реализовать вычисление чисел Фибоначчи через while. Выполнить ручную трассировку. Выполнить трассировку в VS.