Язык C. Базовые конструкции и операторы.

ПРИДУМАЛ НАЗВАНИЕ

ЗАБЫЛ ЗАЧЕМ НУЖНА ПЕРЕМЕННАЯ

Подробнее о типах данных:

Тип	Размер в байтах (битах)	Интервал изменения
char	1 (8)	от -128 до 127
unsigned char	1 (8)	от 0 до 255
signed char	1 (8)	от -128 до 127
int	2 (16)	от -32768 до 32767
unsigned int	2 (16)	от 0 до 65535
signed int	2 (16)	от -32768 до 32767
short int	2 (16)	от -32768 до 32767
unsigned short int	2 (16)	от 0 до 65535
signed short int	2 (16)	от -32768 до 32767
long int	4 (32)	от -2147483648 до 2147483647
unsigned long int	4 (32)	от 0 до 4294967295
signed long int	4 (32)	от -2147483648 до 2147483647
float	4 (32)	от 3.4E-38 до 3.4E+38
double	8 (64)	от 1.7E-308 до 1.7E+308
long double	10 (80)	от 3.4E-4932 до 3.4E+4932

Подробнее об операторах:

Знак операции	Назначение операции	Знак операции	Назначение операции
( )	Вызов функции	<	Меньше, чем
!	Логическое отрицание	<=	Меньше или равно
-	Изменение знака	>	Больше, чем
++	Увеличение на единицу	>=	Больше или равно
--	Уменьшение на единицу	= =	Равно
(тип)	Преобразование типа (т.е. (float) a)	!=	Не равно
*	Умножение	&&	Логическое "И"
/	Деление	||	Логическое "ИЛИ"
%	Определение остатка от деления	?:	Условная (тернарная) операция
+	Сложение	=	Присваивание
-	Вычитание	"+=, - =, *=, /=, %="	Составные операции присваивания (например, a *= b)

Условный оператор:

If-else классический:

```
int value = 1;

if(value == 1)
{
    printf("one");
}
else
{
    printf("not one");
}
```

Цепочка if-else-if:

```
int value = 1;

if(value == 1)
{
    printf("one");
}
else if(value == 2)
{
    printf("two");
}
else if(value == 3)
{
    printf("three");
}
else
{
    printf("not one, two, three...");
}
```

Переключатель (switch):

```
int value = 2;

switch(value)
{
    case 1:
        printf("one");
        break;
    case 2:
        printf("two");
        break;
    case 3:
        printf("three");
        break;
    default:
        printf("not one, two, three...");
        break;
}
```

Циклические конструкции:

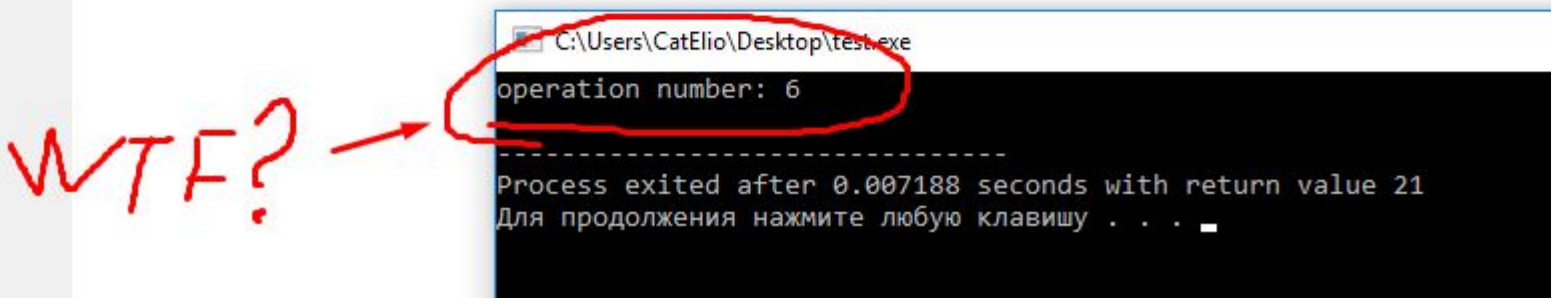
Цикл while:

```
int i = 0;
while(i < 5)
{
    printf("operation number: %d \n", i);
    i = i + 1;
}
```

```
operation number: 0
operation number: 1
operation number: 2
operation number: 3
operation number: 4

--------------------------------
Process exited after 0.009392 seconds with return value 21
Для продолжения нажмите любую клавишу . . .
```

Цикл do-while:

```c
#include <stdio.h>

void main()
{
    int i = 6;

    do
    {
        printf("operation number: %d \n", i);
        i = i + 1;
    }while(i < 5);
}
```



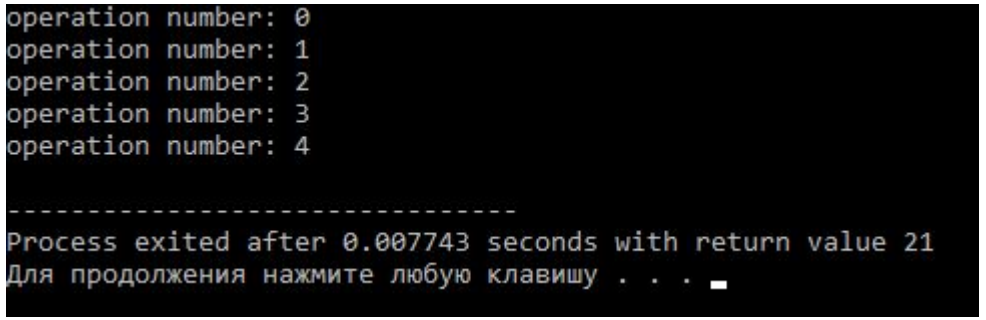
WTF?

Но вернёмся к циклу while:

```
int i = 0;
while(i < 5)
{
    printf("operation number: %d \n", i);
    i = i + 1;
}
```

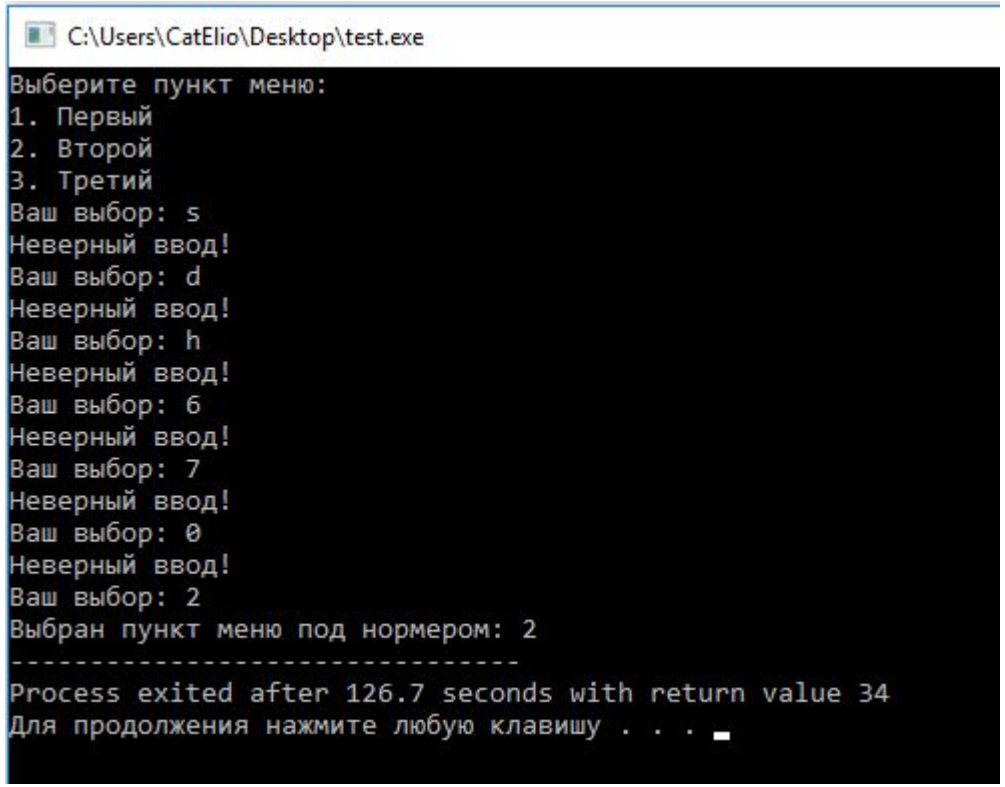
Цикл for:

```
int i;
for(i = 0; i < 5; i++)
{
    printf("operation number: %d \n", i);
}
```



*Кстати, интересный пример на **do-while**:*

```c
#include <stdio.h>
#include <locale.h>
#include <conio.h>

void main()
{

    setlocale(LC_ALL, "Russian");

    char i = 0;

    printf("Выберите пункт меню: \n1. Первый\n2. Второй\n3. Третий\n");

    do
    {
        if(i != 0)
            printf("\nНеверный ввод!\n");

        printf("Ваш выбор: ");
        i = getche();
    }while(i < '1' || i > '3');

    printf("\nВыбран пункт меню под нормером: %c", i);
}
```



Однако, некоторые строки в этом примере нам пока не очень знакомы:

- ‘\n’
- i < ‘1’ || i > ‘3’
- i = getche();

Управляющие последовательности:

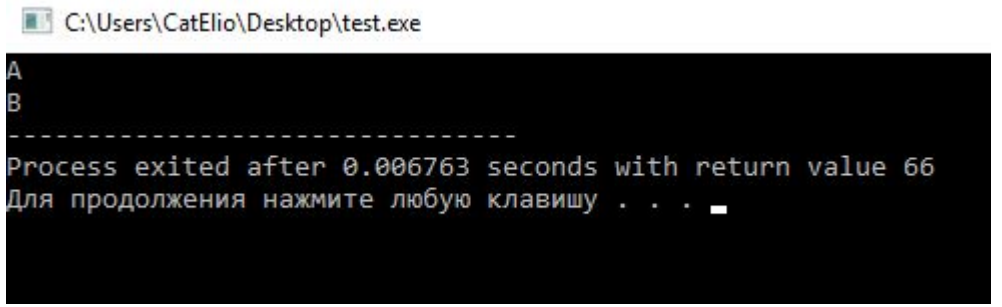
Последовательность	Числовое значение	Результат
\a	0x07	Звуковой сигнал
\b	0x08	Перевод каретки на одно значение назад
\a	0x07	Звуковой сигнал
\f	0x0c	Новая страница
\n	0x0a	Звуковой сигнал
\r	0x0d	Возврат каретки
\t	0x09	Табуляция
\v	0x0b	Вертикальная табуляция
\"	0x22	Двойная кавычка
\\	0x5c	Обратный слеш

Каждый char-символ имеет соответствие в таблице символов ASCII, причём всего их 256.

Таким образом, char по сути является всего лишь уменьшенным int'ом с возможностью кодировки символов.

А это значит, что с ним можно производить все те же действия, что и с int-значениями.

```c
#include <stdio.h>
#include <locale.h>
#include <conio.h>

void main()
{
    char c = 65;
    printf("%c \n", c);

    c = c + 1;
    printf("%c", c);
}
```



Вывести алфавит:

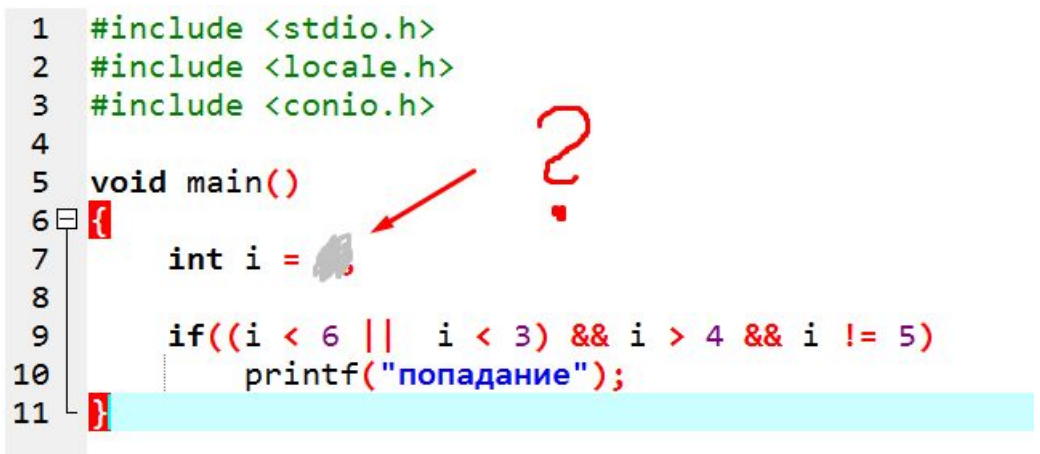
```c
#include <stdio.h>
#include <locale.h>
#include <conio.h>

void main()
{
    char i;

    for(i = 65; i <= 90; i++)
    {
        printf("%c", i);
    }
}
```

```
C:\Users\CatElio\Desktop\test.exe
ABCDEFGHIJKLMNOPQRSTUVWXYZ
--------------------------------
Process exited after 0.01027 seconds with return value 91
Для продолжения нажмите любую клавишу . . .
```

Логические операции:

И	**&&**	a == 3 **&&** b > 4	Составное условие истинно, если истинны оба простых условия
ИЛИ	**||**	a == 3 **||** b > 4	Составное условие истинно, если истинно, хотя бы одно из простых условий
НЕ	**!**	!(a == 3)	Условие истинно, если a не равно 3

И сразу задачка: при каком значении i выведется слово «попадание»?



Ответ: ***ни при каком!***

```c
#include <stdio.h>
#include <locale.h>
#include <conio.h>

void main()
{
    setlocale(LC_ALL, "Russian");
    int i;
    for(i = -10; i < 10; i++)
    {
        if((i < 6 ||  i < 3) && i > 4 && i != 5)
            printf("попадание\n");
        else printf("не попал :(\n");
    }

}
```

```
C:\Users\CatElio\Desktop\test.exe
не попал :(
не попал :(
не попал :(
не попал :(
не попал :(
не попал :(
не попал :(
не попал :(
не попал :(
не попал :(
не попал :(
не попал :(
не попал :(
не попал :(
не попал :(
не попал :(
не попал :(
не попал :(
не попал :(

--------------------------------
Process exited after 0.01892 seconds with return value 0
Для продолжения нажмите любую клавишу . . .
```

Функции:

- *Объявление;*
- *Вызов;*

Пример ***объявления****:*

```
int summ(int a, int b)
{
    int result = a + b;
    return result;
}
```

- *Возвращаемое значение;*
- *Название;*
- *Параметры (аргументы);*
- *Тело функции;*
- *Возврат результата;*

Пример ***вызова****:*

```
void main()
{
    int result = summ(3, 5);

    printf("%d", result);
}
```

- *Возвращаемое значение;*
- *Название;*
- *Параметры (аргументы);*

Примеры часто используемых функций:

int printf(const char *format, arg-list)

Принимает несколько аргументов.

- *Первый: строка-формат вывода;*
- *Последующие – выводимые значения;*

```c
#include <stdio.h>
#include <locale.h>
#include <conio.h>

void main()
{
    int value = 7;
    printf("Value is: %d", 7);
}
```

```
C:\Users\CatElio\Desktop\test.exe
Value is: 7
--------------------------------
Process exited after 0.005837 seconds with return value 11
Для продолжения нажмите любую клавишу . . .
```

%c	Символ типа char
%d	Десятичное число целого типа со знаком
%i	Десятичное число целого типа со знаком
%e	Научная нотация (e нижнего регистра)
%E	Научная нотация (E верхнего регистра)
%f	Десятичное число с плавающей точкой
%g	Использует код %e или %f — тот из них, который короче (при использовании %g используется e нижнего регистра)
%G	Использует код %E или %f — тот из них, который короче (при использовании %G используется E верхнего регистра)
%o	Восьмеричное целое число без знака
%s	Строка символов
%u	Десятичное число целого типа без знака
%x	Шестнадцатиричное целое число без знака (буквы нижнего регистра)
%X	Шестнадцатиричное целое число без знака (буквы верхнего регистра)
%p	Выводит на экран значение указателя
%n	Ассоциированный аргумент — это указатель на переменную целого типа, в которую помещено количество символов, записанных на данный момент
%%	Выводит символ %

int scanf(const char *format, arg-list)

Принимает несколько аргументов.

- *Первый: строка-формат ввода;*
- *Последующие – вводимые значения;*

```c
#include <stdio.h>
#include <locale.h>
#include <conio.h>

void main()
{
    int value1 = 0, value2 = 0;
    scanf("%d%d",
          &value1, &value2);
    printf("Values:\nvalue1: %d\nvalue2: %d",
          value1, value2);
}
```

%c	Считать один символ
%d	Считать десятичное число целого типа
%i	Считать десятичное число целого типа
%e	Считать число с плавающей запятой
%f	Считать число с плавающей запятой
%g	Считать число с плавающей запятой
%o	Считать восьмеричное число
%s	Считать строку
%x	Считать шестнадцатиричное число
%p	Считать указатель
%n	Принимает целое значение, равное количеству считанных до текущего момента символов
%u	Считывает беззнаковое целое
%[]	Просматривает набор символов
%%	Считывает символ %

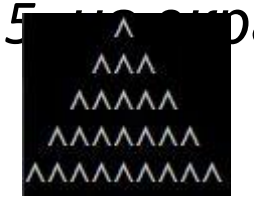
А сейчас несколько задач:

1. Пользователь вводит порядковый номер пальца руки. Необходимо показать его название на экран.

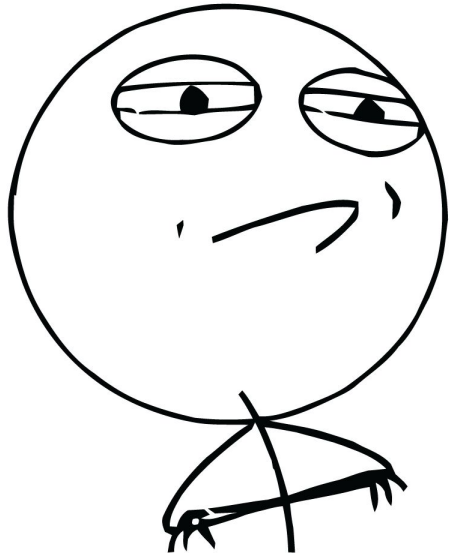
2. Написать функцию, которая будет возвращать квадрат числа, введенного пользователем. Организовать вывод на экран.

3. Напишите свою функцию pow(float basis, int exp) ☺️
Для справки: pow возводит число в степень.

4. Нарисовать равнобедренный треугольник из символов ^. Высоту выбирает пользователь. Например: высота = 5 на экране

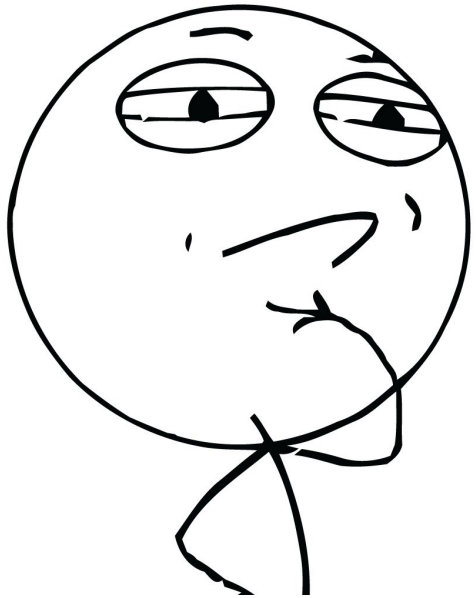


CHALLENGE ACCEPTED



До прочтения условия

CHALLENGE CONSIDERED



После