Кафедра радиоэлектроники

СПРАВОЧНЫЕ МАТЕРИАЛЫ

к лабораторным работам практикума

"Технические средства автоматизации научных исследований"

(ТСАНИ)

ТСАНИ - 0

1992 г.

Справочные материалы к лабораторным работам практикума "Технические средства автоматизации научных исследований" содержат:
- краткую сводку ключевых слов, операций и операторов языка Турбо СИ
- краткое описание необходимого минимума библиотечных функций
- описание функций для работы с КАМАК
- краткая сводка меню редактора интегрированной среды Турбо Си.

Использование этих материалов сразу возможно после прочтения (необходимо, но вполне недостаточно) какой нибудь подробной книги по программированию на языке СИ.

ЛИТЕРАТУРА.

Учебники по СИ.

1. Б. В. Керниган, Д. Ритчи, Д. Фьюэр. Язык программирования Си.
   М.: Финансы и статистика, 1985.

2. Р. Уинер. Язык ТУРБО СИ.
   М.: Мир, 1991.

3. М. Уэйт, С. Прата, Д. Мартин. Язык СИ. Руководство для начинающих.

4. С.В. Бочков, А.М. Субботин. Язык программирования СИ для персонального компьютера. М.; Радио и связь, 1990.

Для работы с КАМАК:

4. ТСАМИ - 2 (Методические указания к работе №2).

Составитель: Колесников К. А.

ТСАМИ - 0 1992 г.

Ключевые слова ТУРБО СИ.

break	default	enum	goto	return	switch	void
case	do	extern	if	sizeof	typedef	while
char	double	float	int	static	union	
continue	else	for	long	struct	unsigned	

Комментарии заключаются символами /* и */.

Оператор - выражение, после которого следует точка с запятой (;).

Блок операторов.
Группа операторов заключенная в фигурные скобки ({, }).
Тело функции также заключается в фигурные скобки.

```
extern      спецификаторы класса памяти
static
        extern char* ptr;       по умолчанию инициализируются значением
        static int beta = 5;    нуль.
        static float pi;
```

```
if      else            управляющие структуры
switch  case    default
do      while
for
```

```
continue                прерывание цикла
break
```

Оператор break внутри цикла приводит к немедленному выходу из цикла и передаче управления на следующий за циклом оператор.
Оператор continue приводит к прерыванию цикла и переходу в то место где производилась проверка условия цикла.

```
if ( выражение ) оператор (или блок);
   или
if ( выражение ) оператор1 (или блок);
else оператор2 (или блок);
```

Если выражение в заголовке условного оператора вырабатывает ненулевое значение (выражение истинно), то оператор в условном операторе выполняется, в противном случае управление передается оператору, следующему за условным, или выполняется оператор следующий после слова else (если оно имеется).

```
if (d > 0)
 {
   printf(" d=%d",d);
   d++;
 }
else printf(" d < 0");
```

Арифметическое if (? :).

```
   результат = выражение ? выражение1 : выражение2 ;

Например:    a = ( i < j) ? i : j;
```

переменной a будет присвоено значение i, если выражение (i < j) истинно, и j, если выражение (i < j) ложно.

```
switch (выражение)  /* вир.. - целого типа */
{
  case константное_выражение_1 :
                                оператор(и);
  case константное_выражение_2 :
                                оператор(и);
  case константное_выражение_3 :
                                оператор(и);
       ...
                        default:
                                оператор(и);
}
```

После вычисления выражения в заголовке оператора его результат последовательно сравнивается с выражениями, начиная с самого верхнего, пока не будет установлено их соответствие. Тогда выполняются операторы внутри соответствующего case.
Ветка default (умолчание) может отсутствовать.
После входления в данный case операторы будут выполняться до первого оператора break или до конца оператора switch (т.е. будут выполнены операторы входящие в другие case'и, легащие ниже точки входления в данный case).

```
#include <conio.h>
#include <stdio.h>
void main() /* вывожим на экран код */
{           /* нажатой клавиши */
 int key;
 while( (key = getch()) != 27)
 {
  switch( key )
  {
    case 0 : getch();
             printf("\nСкан - код клавиши %d",
                                        key);
             break;
   default : printf("\n ASCII - код %d", key);
             break;
  }
 }
  printf(" Была нажата клавиша ESC , код = %d", key);
}
```

```
while ( условное_выражение) оператор;
```

Если результат вычисления условного выражения не равен нулю, то выполняется оператор или группа операторов (блок). Условие проверяется вначале, затем выполняется тело цикла.

```
int a; char* b;
...
while (a != 0 && *b !=0)
  {
     printf("\na=%d   b=%c", a, *b);
     a--;
     b++;
  }
```

```
do
   оператор;
while( условное_выражение);
```

```
int a=0;
do  printf("\n%d",a);
while ( ++a <= 100);
```

/* условие проверяется после выполнения тела цикла */

```
for( необязательное_вир1; необяз_вир2; необяз_вир3)
                                оператор;

float r, k;
int i;
for (r=1.2, k=0.5; r >= -0.1 && k < 100.; r -= 0.12,
                                    k *= 2.1)
 {
```

Каждое из трех выражений можно опускать. Тогда в принципе каждое из выражений может быть использовано как угодно, обычно первое выражение служит для инициализации счетчика


```
   if(r == 0) continue;
   printf("\nчастное k/r = %f", r/k);
 }
```

цикла, второе - для выполнения проверки на окончание цикла, а третье выражение - для изменения значения счетчика цикла.

```
goto label;
...
label : оператор;
```

Оператор goto используется для передачи управления внутри функции от одного оператора к другому.

char	struct	основные типы	Простой тип (константа) может быть использован в СИ программе, его можно быть описано.
int	union		
long			
unsigned			
enum	typedef	возможно определение нового типа заданного сами пользователем	
float			
double			
void			

Описания имен должны быть первыми в теле функции или блоке, ограниченными скобками { и }.

```
char ch;     /* переменная ch типа char */
int count=1;/* переменная count типа int, начальное значение=1 */
long int k, ti;
double beta=1.67823;
```

Прототипы функций:

```
void f( int, float );  /* функция f со параметров типа int и float,
                          не возвращает значение */
int ft( void );/* а функция ft не передается ни какие
                  параметры, возвращает значение типа int  */
```

Прототипы помещаются перед телом main() или помещаются в отдельные файлы с расширением .h (хедер-файл) и подключаются с помощью #include.

Описание тела функции:

```
void f(int s, float x1)
   {
       int i;
       for( i=1; i<10; i++) printf("\nt=%4d  x=%f", i*s, x1/i);
   }

   int Getch(void)
   {
          int a;
          if( (a = getch()) != 0) return a;
          else return (getch()+255);
   }
```

ВЕКТОРА И УКАЗАТЕЛИ

```
int g[5];/*  g[0], g[1], g[2], g[3], g[4] элементы массива g */
int d[2] [3] ;/* d[0][0], d[0][1], d[0][2], d[1][0], d[1][1], d[1][2]*/
float point[20];  /* массив из 20 элементов типа float */
```

Инициализация
```
int v[] = {1, 2, 3, 4, 5};  /* в таком виде компилятор сам вычисляет размер*/
char buf[] = "Это строка."; /* размер на 1 больше, компилятор добавит
                               символ '\0' - конец строки.*/
```

```
char* string="Строка символов"; /* указатель на строку  */
int* pi;   /*   pi указатель на int    */
char** cpp; /*    cpp указатель на указатель на char  */

int (*lab) [10];/* lab указатель на вектор из 10 int-ов  */
```

Перечисления
```
enum { ASM, AUTO, BREAK }; /*  Это значит, что ASM=0,  AUTO=1, BREAK=2.*/
enum example { a = 10, b = -5, c=15, d, e=10 }; /* значение d = 16 !
                                                   Значения перечислителей не
                                                   обязательно должны быть
                                                   различными, возрастающими
                                                   или положительными.*/
```

Структуры и объединения struct, union

```
struct address {    /* почтовый адрес */
char* name;  /*   имя        */
long number; /*  номер дома   */
char* street;/*  название улицы */
char* town;  /*  название города*/
char state[2];/* штат          */
int zip;      /* индекс        */
};   /* точка с запятой обязательна   */

struct address jd = {                          Инициализация
    "Jim Dandy",
    61, "Fourth Street",
    "New York", {'N','J'}, 7974
    };

struct address* p;/* p указатель на структуру типа   ОБРАЩЕНИЕ к членам
                     address*/                       с помощью указателя.
p->name = "Jim Dandy";
p->number = 61;
```

```
union alternative_data          Используя объединения (union), можно в одной
{                             и той же области размещать данные различных типов.
char s[4];                    Естественно, что  в данный момент времени в памяти
float r;                      могут быть  размещены значения только одного вклю-
int i;                        ченного в объединение типа. Размер памяти, требуе-
char ch;                      мой для  размещения  объединения определяется раз-
unsigned j;                   мером наибольшего поля.
};
```

```
typedef unsigned int CARDINAL; /* определяется новый тип CARDINAL как
                                  unsigned int */
CARDINAL n; /* переменная n типа CARDINAL  (unsigned int)*/
```
---

char	1 байт
int	2 байта
short int	2 байта
long int	4 байта
float	4 байта
double	8 байт

диапазон принимаемых значений	
char	-128...127
unsigned char	0...255
int	-32768... 32767
unsigned int	0...65535
long int	-2147483648... ...2147483647
unsigned long	0...4294967295
float	+- 3.4E+-38
double	+- 1.7E+-308

sizeof операция определения размера отведенной памяти под тип или объект.

return; возврат из функции
return x; возвращение значения x при возврате из функции

Краткая сводка операций С.
  В каждой очерченной части находятся операции с одинаковым приоритетом. Операции, находящиеся в верхней части таблицы имеют больший приоритет, чем операции, находящиеся в нижней части.
    Например: a+b*c означает a+(b*c).

  Унарные операции и операции присваивания правоассоциативны, все остальные левоассоциативны. Это значит, что a=b=c означает a=(b=c), a+b+c означает (a+b)+c, и *p++ означает *(p++), а не (*p)++.

->	выбор члена структуры	указатель->член
[]	индексация	указатель [выражение]
()	вызов функции	выражение (список_выр)
sizeof	размер объекта	sizeof (выражение)
sizeof	размер типа	sizeof (тип)
++	инкремент постфикс (после)	lvalue++
++	инкремент префикс (до)	++lvalue
--	декремент после	lvalue--
--	декремент до	--lvalue
!	логическое не	! выражение
*	не (побитовое)	* выр
-	унарный минус	- выр
+	унарный плюс	+ выр
&	адрес объекта	& lvalue
*	разыменовывание указателя	* выр
()	приведение (преобразование типа)	(тип) выражение

*	умножение	выр1 * выр2
/	деление	выр1 / выр2
%	взятие по модулю (остаток)	выр1 % выр2
+	сложение	выр1 + выр2
-	вычитание	выр1 - выр2
<<	сдвиг влево	lvalue << выр
>>	сдвиг вправо	lvalue >> выр
<	меньше	выр1 < выр2
<=	меньше или равно	выр1 <= выр2
>	больше	выр1 > выр2
>=	больше или равно	выр1 >= выр2
==	равно	выр1 == выр2
!=	не равно	выр1 != выр2
&	побитовое И	выр1 & выр2
^	побитовое исключающее ИЛИ	выр1 ^ выр2
|	побитовое ИЛИ	выр1 | выр2
&&	логическое И	выр1 && выр2
||	логическое ИЛИ	выр1 || выр2
? :	арифметический if	выр1 ? выр2 : выр3
=	простое присваивание	lvalue = выр
*=	умножить и присвоить	lvalue *= выр
/=	разделить и присвоить	lvalue /= выр
%=	взять остаток и присвоить	lvalue %= выр
+=	сложить и присвоить	lvalue += выр
-=	вычесть и присвоить	lvalue -= выр
<<=	сдвинуть влево и присвоить	lvalue <<= выр
>>=	сдвинуть вправо и присвоить	lvalue >>= выр
&=	И и присвоить	lvalue &= выр
|=	ИЛИ и присвоить	lvalue |= выр
^=	исключающее ИЛИ и присвоить	lvalue ^= выр
,	запятая (слепование)	выр1, выр2

Операции совпадения результат применения: истина это 1 ложь - 0.

a+=5 эквивалентно a=a+5

ПОБИТОВЫЕ операции И, ИЛИ, исключающее ИЛИ

И (&)	0	1
0	0	0
1	0	1

ИЛИ (|)	0	1
0	0	1
1	1	1

искл. ИЛИ (^)	0	1
0	0	1
1	1	0

При операции ~ каждый нулевой бит заменяется на 1, а каждая единичка становится нулем (побитовая инверсия).

Командам препроцессору:

```
  #define   определение макро.
  #include  подстановка текста из внешнего файла.

#include <dos.h>   /* подстановка файла из директории INCLUDE среды TC3/
#include <stdio.h>
#include "prob.h"  /* подстановка файла из текущей директории */
#include "fital.c"

#define BEGIN {
#define END }
#define PI 3.14159265358979323385
     Когда BEGIN, END или PI встречаются как лексемы, то они заменяются
соответственно на  {, },  и 3.141592...
   Можно также определить макрос с параметрами:

#define MIN(a,b) (((a)<(b))?(a):(b))
```

КОНСТАНТЫ

```
  0356      - "восьмеричная"          форма записи целых констант
 -15        - "десятичные"
  1234152L  - константа типа long int
  3678U     - константа типа unsigned int
  0x33fe    - "шестнадцатиричные"
  0XA299

 1.23456  2566.  -.2002345    1.56-E13  - вид записи вещественных
```

Символьная константа - это символьный символ, заключенный в апострофы.

например: 'x', 'a', 'H'

Некоторые не имеющие графического представления символы, а также апостроф 'и обратная косая черта \ , представляются в виде управляющих последовательностей (escape-последовательностей) в соответствии со следующей таблицей:

новая строка	NL(LF)	\n
горизонтальная табуляция	HT	\t
вертикальная табуляция	VT	\v
возврат на одну позицию	BS	\b
возврат каретки	CR	\r
прогон страницы	FF	\f
обратная косая черта	\	\\
апостроф	'	\'
двойная кавычка	"	\"
звонок	BELL	\a
нулевой символ	NULL	\0
битовый шаблон	0ddd	\ddd
	0xddd	\xddd

Управляющая последовательность \ddd состоит из обратной косой черты, за которой следуют 1, 2 или 3 восьмеричных цифры, задающие значение желаемого символа ('\101'). 0xddd - цифры шестнадцатиричные.

ПРЕДСТАВЛЕНИЯ ЧИСЕЛ

Если переменная описана как unsigned, это значит, что старший разряд значения переменной (в двоичном представлении) не воспринимается как знаковый.

Например: значение 1111111111111111 воспринимается как -1 для int, и 65535 для unsigned int.

Отрицательные целые числа представляются в дополнительном коде:

```
1000000000000000 это -32767
1111111111111111     -1
```

положительные в прямом коде:

```
111111111111111 это  32767
              1      1
```

Старший разряд (для int 16-ти) воспринимается как знаковый.
Аналогично для char и long.

СПРАВОЧНАЯ ИНФОРМАЦИЯ ПО БИБЛИОТЕЧНЫМ ФУНКЦИЯМ Си.

Некоторые функции вывода (ввода) на стандартное устройство (stdout (stdin)) .

По умолчанию стандартное устройство вывода - экран дисплея. Стандартное устройство ввода - клавиатура.

```
DOS_PROMT> program.exe > file_name    перенаправление вывода в файл с именем
                                      file_name.

DOS_PROMT> program.exe < file_name    ввод будет осуществляться из файла
                                      с именем file_name.
```

Значение EOF (End Of File) равно -1.

Прототипы в stdio.h #include <stdio.h>

```
int puts (const char* string);      вывода строки на устройство вывода
char* gets (char* string);          ввода с устройства ввода

include <stdio.h>                       ввод заканчивается по обнаружении
void main()                             символа '\n' (клавиша <ENTER>)
{
    char buf [80];                      при выводе автоматически добавляется
    puts ("\n Input a string: ");       '\n' (перевод строки).
    gets (buf);
    puts (buf);
}
```

```
int printf( const char* format [, argument, ...]);   форматированный вывод
int scanf (const char* format, address, ...);        форматированный ввод
```



```
include <stdio.h>
void main()
 {
    int delta;
    double tera;
    puts ("\nВведите координаты через пробел: ");
    scanf("%5d %2.5lf", &delta, &tera);
    printf("\n delta = %d\n tera = %lf", delta, tera);
}
```

```
int sprintf (char* buffer, const char* format [, argument, ...]);
int sscanf (const char* buffer, const char* format [, address, ...]);
```

Форматированный вывода в строку и ввода из строки соответственно.

```
include <stdio.h>
void main()
{
    double a1, a2, a3;
    char buf[80];
    char n1[] = "13.25 10 21.789";
    sscanf(n1, "%lf %lf %lf", &a1, &a2, &a3);
    sprintf(buf, "\n a1 = %lf\n a2 = %lf\n a3 = %lf", a1, a2, a3);
    puts(buf);
}
```

```
int putchar (int ch);       вывода символа
int getchar (void);         ввода символа, ввод заканчивается при обнаружении
                            '\n'. Возвращает последний введенный символ.

include <stdio.h>
void main()
{
    int c;
    while(( c = getchar()) != '\n') putchar(c); /* после окончания ввода
}                                  последовательно выводятся символы из буфера*/
```

format

Спецификация формата для ввода или вывода выглядит следующим образом:

```
  % [flags] [width] [.prec] [F|N|h|l] type
```

Спецификатор type

type	Формат ввода - вывода
d	signed decimal int целое в десятичной системе счисления
i	signed decimal int
o	unsigned octal int целое в восьмеричном виде
u	unsigned decimal int беззнаковое целое в десятичной системе
x	printf = unsigned hexadecimal int целое в шестнадцатиричной форме
	scanf = hexadecimal int
X	printf = unsigned hexadecimal int;
	scanf = hexadecimal long

f	Вывод (ввод) чисел с плавающей запятой Floating point [-]dddd.ddd
e	Floating point [-]d.ddd e [+/-]ddd (в экспоненциальной форме)
g	использует формат e или f в зависимости от точности
E	то же, что и e за исключением E для обозначения экспоненты
G	то же, что и g за исключением E для обозначения экспоненты
c	Один символ
s	Ввод (вод) строки до символа '\0'(конец строки) или в соответствии с [.prec]
%	знак %

type в строке формата должен соответствовать типу аргумента, значение которого выводится или куда предполагается помещать вводимую информацию.

Спецификация формата "[F|N|h|l]" (| - или)
Используется совместно с type

Модификатор	как будет интерпретирован аргумент
F	far указатель
N	near указатель
h	d,i,o,u,x,X аргумента типа short int
l	d,i,o,u,x,X аргумента типа long int
l	e,E,f,g,G аргумента типа double (только в scanf)
L	e,E,f,g,G аргумента типа long double

Спецификатор формата "[.prec]"

(Количество знаков после точки)

[.prec]	влияние на вывод
(без)	точность (количество значащих цифр) по умолчанию
.0	для (d,i,o,u,x) точность по умолчанию для (e,E,f) отсутствует десятичная точка
.n	не более n знаков
*	Следующий аргумент списка = точность

Спецификатор формата "[width]"

(количество позиций под выводимое число)

[width]	влияние на вывод
n	выводится не более n символов
0n	выводится n символов, незначащие знаки выводятся как нули слева
*	следующий аргумент списка - width

Спецификатор формата "[flag]"

[flag]	влияние на вывод
(без)	выравнивается по правому краю, слева дополняется нулями или пробелами
-	выравнивается по левому краю, дополняется пробелами справа
+	всегда выводится знак (+ или -)
пробел	знак выводится только для отрицательных величин

НЕКОТОРЫЕ ПОЛЕЗНЫЕ ФУНКЦИИ

```
int getch(void);      принимает символ с клавиатуры без эха
int getche(void);                                  с эхом

include <conio.h>
void main()
{
     while (getch() != 'y') /* опрашивает ввода и выполняет тело цикла, если
                               не нажата клавиша <y>*/
     puts(" Продолжить ? (y/n)");
     puts(" Продолжили...");
}
```

```
int kbhit(void);      Возвращаемое значение - 0, если буфер  клавиатуры
                      пуст. Возвращает не ноль, если была нажата клавиша (в
                      буфере клавиатуры что-то есть). Символ из буфера можно
include <conio.h>           извлечь с помощью функции getch().
void main()
{
    while( kbhit() == 0) puts("$");
    printf("\nКод нажатой клавиши = %d", getch());
}
```

```
void clrscr(void);              очищает экран в текстовом моде
clrscr();
```

ГРАФИЧЕСКАЯ МОДА ДИСПЛЕЯ

Прототипы всех описываемых функций в graphics.h

```
#include <graphics.h>
```

```
Инициация с диагностикой.
     initgraph( &GraphDriver, &GraphMode, "PATH_TO_DRIVER" );
        if( ( ErrorCode = graphresult() ) != grOk ) {
                printf(" Graphics System Error: %s\n",
                        grapherrormsg( ErrorCode ) );
                exit( 1 );
        }

int  GraphDriver = EGA;
int  GraphMode  = EGAHi = 1;  640x350 16 цветов 2 страницы;
```

```
PATH_TO DRIVER, например, N:\\BC3\\BGI - каталог, где находится программа
                драйвер (egavga.bgi для дисплеев типа EGA и VGA).
-------------------------------------------------------------------------
Информация.
    maxcolor = getmaxcolor ();      максимальный цвет в данной моде
    maxx = getmaxx ();              максимальная координата по x
    maxy = getmaxy ();              максимальная координата по y
-------------------------------------------------------------------------
    restorecrtmode ();          Возврат в текстовую моду.

    setgraphmode( getgraphmode () );  Возврат из текстовой моды в
                                      пред. графическую. Все чистится.

    graphdefault ();   Установка default-параметров (по умолчанию).

    closegraph ();     Закрыть графику, выгрузить драйвер.
-------------------------------------------------------------------------
    cleardevice ();    Чистка экрана.
-------------------------------------------------------------------------
Нарисовать точку, взять цвет точки.          Начало отсчета координат:
    putpixel ( x, y, color );                верхний левый угол экрана
    color = getpixel ( x, y );

Цвет линий, текста, окружностей и т.п.
    setcolor ( color );             установить цвет
    color = getcolor ();            узнать текущий цвет
Цвет фона
    setbkcolor ( color );           установить
    color = getbkcolor ();          узнать текущий

Цвет и тип заполнения для bar, bar3d.    fill = 0 = EMPTY_FILL - цветом фона;
    setfillstyle ( fill, color );              1 = SOLID_FILL - цветом color;

Тип линии для line, circle, bar3d
    setlinestyle ( style, 0, width );
        style = 0 = SOLID_LINE   - сплошная;
                1 = DOTTED_LINE  - точками;
                3 = DASHED_LINE  - пунктир;
        widths = 1 = NORM_WIDTH  - нормальная;
                 3 = THICK_WIDTH - толстая;

Провести линию.
    line ( x1, y1, x2, y2 );  отрезок точками
    linerel ( dx, dy );       от текущего положения курсора в точку с
                              относительными координатами
    lineto   ( x, y );        от текущего положения курсора в точку с
                              с абсолютными координатами

Установить параметры вывода текста.
    settextstyle ( font, direction, charsize );
        font = 0 = DEFAULT_FONT;   Шрифт 8x8 обычный.
        direction = 0 = HORIZ_DIR - горизонтально.
                    1 = VERT_DIR  - вертикально.
        charsize = 1...           - увеличен размер.

        default = settextstyle ( 0, 0, 1 );
```



```
Куда выводится текст относительно курсора.
   settextjustify ( x, y );
          курсор находится:
          x = 0 = LEFT_TEXT    - слева от текста.
              1 = CENTER_TEXT  - в центре текста.
              2 = RIGHT_TEXT   - справа от текста.
          y = 0 = BOTTOM_TEXT  - снизу от текста.
              1 = CENTER_TEXT  - в центре текста.
              2 = TOP_TEXT     - сверху от текста.

    default = settextjustify ( 0, 2 );

   Курсор двигается при выводе текста только для  x = 0;

Поставить курсор, взять текущее положение курсора.
    moveto    ( x, y );      абсолютные координаты
    moverel ( dx, dy );      координаты отсчитываются от текущих
    x = getx ();
    y = gety ();

Вывод текста.
    outtext   ( "text" );          относительно текущего положения курсора
    outtextxy ( x, y, "text" );

Прочие картинки.
    rectangle( left, top, right, bottom);  прямоугольник
    bar ( left, top, right, bottom);       прямоугольник с заполнением

    bar3d ( left, top, right, bottom, depth, topflag);  параллелепипед
    circle ( x, y, radius);                             окружность
    arc ( x, y, start_angle, end_angle, radius);          дуга
    pieslice ( x, y, start_angle, end_angle, radius);     сектор
    ellipse ( x, y, start_angle, end_angle, x_radius, y_radius);  эллипс
-------------------------------------------------------------------------
    Стандартный набор цветов ( параметр color ):

    BLACK     == 0     DARKGRAY     == 8
    BLUE      == 1     LIGHTBLUE    == 9
    GREEN     == 2     LIGHTGREEN   == 10
    CYAN      == 3     LIGHTCYAN    == 11
    RED       == 4     LIGHTRED     == 12
    MAGENTA   == 5     LIGHTMAGENTA == 13
    BROWN     == 6     YELLOW       == 14
    LIGHTGRAY == 7     WHITE        == 15

РАБОТА С ПОРТАМИ ВНЕШНИХ УСТРОЙСТВ
=========================================================================
Провозимы в dos.h  #include <dos.h>
-------------------------------------------------------------------------
 int inp (int portid);              считывает байт из порта с адресом portid.
 unsigned char inportb(int portid);

 c = inp(0x61);
-------------------------------------------------------------------------
 int outp (int portid, int byte_value);              пишет байт в порт.
 void outportb (int portid, unsigned char value);
```

```
outp (0x241, data);
```

Другие полезные функции.

```
void sound (unsigned frequency);   PC начинает издавать звук с частотой
                                   frequency ( в герцах).
void nosound (void);       выключает звук.

#include <dos.h>
void main()
{
  sound (1000);
  delay (500);
  nosound();
}
```

```
void delay (unsigned milliseconds);  задерживает программу на некоторое
                                     время ( в мсек).

void sleep (unsigned seconds);       задерживает программу на некоторое
                                     время ( в сек).
```

```
void gettime (struct time *timep);      можно узнать системное время.

 struct time {                      #include <dos.h>
   unsigned char  ti_min;           #include <stdio.h>
   unsigned char  ti_hour;          void main() {
   unsigned char  ti_hund;          struct time t;
   unsigned char  ti_sec;           gettime (&t);
 };                                 printf ("Time is %2d:%02d:%02d.%02d",
                                            t.ti_hour, t.ti_min, t.ti_sec,
                                            t.ti_hund);
                                    }
```

РАБОТА С КАМАК

Адреса портов диспетчера:

```
0x241 средний       байты
0x242 младший       для записи
0x243 субадрес в блоке ( A )
0x244 функция          ( F )
0x245 номер блока      ( N )
0x246 генерация  C,Z
0x247 инициация обмена
0x248 ответ L,X,Q
0x249
0x24A средний       прочитанные
0x24B младший       байты
0x24C служебный регистр
0x24D
0x24E
0x24F служебный регистр ППИ - 4 - PC
```


Форматы портов :

```
0x241- |N16|N15|N14|N13|N12|N11|N10| N9| средний байт
                                      для записи
0x242- | N8| N7| N6| N5| N4| N3| N2| N1| младший байт
                                      для записи

0x243- |   |   |   |   |  A8|  A4|  A2|  A1|  субадрес

0x244- |   |   |   |F16|  F8|  F4|  F2|  F1|  функция

0x245- |   |   |   |N16|  N8|  N4|  N2|  N1|  номер блока в
                                      крейте

0x246- |   |   |   |   |   |   | C | Z |  команды управления
                                  |   |
                                  |   |-генерация Z при установленном бите
                                  |-генерация C при установленном бите

0x247- |   |   |   |   |   |   |   |   |   при обращении к этому порту
                                      происходит генерация КАМАК - цикла

0x248- | L |   |   |   |   |   | X | Q |  - ответ L,X,Q блока на
                                      последний КАМАК - цикл

0x24A- |R16|R15|R14|R13|R12|R11|R10| R9| средний байт для чтения
0x24B- | R8| R7| R6| R5| R4| R3| R2| R1| младший байт для чтения

0x24C- |   |   |   |   |  ED| ON|  L |  ok|  - служебный  регистр
                           |   |   |   |
                           |   |   |   |--- 1 если последний цикл завершен
                           |   |   |
                           |   |   |-------- 1 если  возникло L
                           |   |
                           |   |------------ 1 если крейт ON LINE
                           |
                           |---------------- 1 конец АМА

0x24D - ????

0x24F - | D7| D6| D5| D4| D3| D2| D1| D0| - служебный регистр диспетчера
                  |   |   |   |   |   |
                  |   |   |   |   |   |----|-- число байт на цикл АМА
                  |   |   |   |   |
                  |   |   |   |   |--------- 1 останов АМА при обнулении Q
                  |   |   |   |
                  |   |   |   |------------- направление передачи АМА
                  |   |   |
                  |   | ---|---------------- номер крейта
                  |---------------------- инициализация  АМА
```

Служебный регистр диспетчера позволяет организовать пересылку данных в режиме прямого доступа к памяти (АМА). В лабораторных работах не рекомендуется использовать этот режим и поэтому в служебном регистре необходимо использовать лишь D4 и D5 (остальные биты равны нулю). Регистр доступен только для записи !!!

Функции для работы с КАМАК-крейтом.

Прототипы в tsani.h #include <tsani.h>

Библиотека tsani.lib предназначена для работы в среде компилятора TC, с моделью памяти LARGE.
Название библиотеки - tsani.lib необходимо указать в файле-проекте. Файл - проект это текстовый файл, в котором указаны имена файлов из которых будет собираться конечный файл с расширением .exe (исполняемый файл).

```
file.prj        my_file1.c
                my_file2.c
                ..........
                tsani.lib
```

Имя проекта должно быть указано в окне PROJECT меню TC.

```
int cam_crate ( int crate);      устанавливает адрес крейта в PPI-PC

int cam_i (int n, int a, int f, unsigned* data);
   Выполняет КАМАК цикл, пишет или читает данные, в зависимости от f:
   если 15 < f < 31 - пишет из &data по адресу na, иначе читает в &data.
   Возвращает L, X, Q от NAF-а в формате: 0000 0000 L000 00XQ.
   L - возникает если есть хоть один забаскированный запрос.

int cam_naf (int n, int a, int f);
   Выполняет КАМАК цикл, возвращает L, X, Q от NAF-а в том же формате что
   и cam_i(). Предназначен для генерации цикла КАМАК без данных.
     Например: N, A(0), F(8).

void cam_setnaf(int n, int a, int f);
   Записывает регистры N, A, F  PPI-PC и больше ничего не делает.
   Предполагает  последующее  выполнение cam_w_data() или cam_r_data().
   При многократном использовании работает быстрее, чем cam_i().

int cam_w_data (unsigned* data);
   Записывает регистры данных PPI-PC из &data и выполняет КАМАК цикл с
   N, A, F, записанными  ранее  при помощи процедуры cam_setnaf().
   Возвращает  L, X, Q  от NAF-а в формате: 0000 0000 L000 00XQ.

int cam_r_data (unsigned* data);
   Выполняет КАМАК  цикл с  N, A, F, записанными ранее при  помощи
   процедуры cam_setnaf(), и читает данные из PPI-PC  в  &data.
   Возвращает L, X, Q от NAF-а в  формате: 0000 0000 L000 00XQ.

int cam_r_hb (void);        Возвращает содержимое регистра старшего
                            байта крейт-контроллера.

void cam_w_hb (int data);   Записывает регистр старшего байта
                            крейт-контроллера.

void cam_z (void);              Выполняет КАМАК цикл ZERO.

void cam_c (void);              Выполняет КАМАК цикл CLEAR.
```



```
void cam_on_i (void);           Устанавливает INHIBIT.

void cam_off_i (void);          Сбрасывает INHIBIT.

int cam_r_mask (void);          Возвращает содержимое регистра маски
                                и запросов крейт-контроллера.

void cam_w_mask (int data);     Пишет маску в регистр маски и запросов
                                крейт-контроллера.

int cam_r_stat (void);
  Возвращает содержимое регистра статуса и управления крейт-контроллера.

void cam_w_stat (int data);
   Записывает регистр статуса и управления крейт-контроллера (то, что
   можно записать).
```

ДЕСЯТЬ ФУНКЦИЙ ДЛЯ ВВОДА ЧИСЕЛ (аналоги scanf(...)).

Прототипы в tsani.h #include <tsani.h>

```
char ini(char* text, int* data);                Вывода текста и текущее
char inu(char* text, unsigned int* data);       значение  переменной.  В
char ino(char* text, unsigned int* data);       ответ на <Enter> значе-
char inx(char* text, unsigned int* data);       ние  переменной  не  меня-
char inl(char* text, long int* data);           ется,  а возвращается ноль.
char inul(char* text, unsigned long int* data); Иначе  возвращается
char inol(char* text, unsigned long int* data); первый  символ  во
char inxl(char* text, unsigned long int* data); введенной строке.
char inf(char* text, float* data);
char ing(char* text, float* data);
```

Если введется не число,то значение переменной не изменяется.
Пример:

```
int data;
key = inul("\n data = ", &data);
if(key == 'q') exit(0); /* program terminated */
```

ФУНКЦИИ ВЫВОДА ЗНАЧЕНИЯ ПЕРЕМЕННЫХ В ГРАФИЧЕСКОМ РЕЖИМЕ
(аналоги функции printf(...)).

Прототипы в tsani.h #include <tsani.h>

```
int gprintf(int x, int y, char* format [, argument, ...]);
int gprintc(int color, int x, int y, char* format [, argument, ...]);
```

Пример:

```
float temp;
setcolor(RED);
gprintf(250,100," Temperature = %f", temp);
gprintc(14, 250, 100, " Temperature = %f", temp);
```

TYPSO CM .
Назначение функциональных клавиш и сводка команд редактора.

------ HELP ----------		помощь
Help	<F1>	
Index	<ShiftF1>	
Topic search	<CtrlF1>	Поиск по теме (выдается справка о функции или операции СИ, на которой находился курсор
Previous topic	<AltF1>	предыдущая тема help.
------ FILE ----------		
Open	<F3>	открыть файл (для загрузки)
Save	<F2>	записать файл на диск
Quit	<Alt-X>	выход в MS DOS.
------ RUN ----------		
Run	<CtrlF9>	запустить программу на счет
Goto cursor	<F4>	
Trace into	<F7>	пошаговое (построчное) исполнение программы, включая подпрограммы.
Step over	<F8>	трассировка программы без захода в подпрограммы.
Program reset	<CtrlF2>	сброс режима трассировки программы
Compile	<AltF9>	скомпилировать текст программы
Make	<F9>	создать .exe файл
------ DEBUG ----------		справка
Inspect	<AltF4>	
Evaluate-modify	<CtrlF4>	модифицирование переменных
Toggle breakpoint	<CtrlF8>	убрать метки останова
Watches	<CtrlF7>	наблюдение за значениями переменных
------ WINDOW ----------		
Zoom	<F5>	расширить окно на весь экран
Next	<F6>	перейти в другое окно
User screen	<AltF5>	окно вывода программы

Сводка команд редактора

Символ влево	<Ctrl-S>, <Left>	курсоры <-
Символ вправо	<Ctrl-D>, <Right>	->
Слово влево	<Ctrl-A>	
Слово вправо	<Ctrl-F>	
Строка вверх	<Ctrl-E>, <UP>	курсоры вверх
Строка вниз	<Ctrl-X>, <DOWN>	вниз
Страница вверх	<Ctrl-R>, <PgUp>	
Страница вниз	<Ctrl-C>, <PgDn>	
Начало строки	<Ctrl-Q> S, <Home>	
Конец строки	<Ctrl-Q> D, <End>	
Верх экрана	<Ctrl-Q> E	
Низ экрана	<Ctrl-Q> I	
Верх файла	<Ctrl-Q> R	
Низ файла	<Ctrl-Q> C	
Начало блока	<Ctrl-Q> B	
Конец блока	<Ctrl-Q> K	

Команды вставки и замены

Режим вставки ON/OFF	<Ctrl-V>, <Ins>
Вставить строку	<Ctrl-N>
Удалить строку	<Ctrl-Y>
Удалить до конца строки	<Ctrl-Q>
Удалить символ слева от курсора	<Ctrl-H>, <Backspase>
Удалить символ в позиции курсора	<Ctrl-G>, <Del>
Удалить слово справа от курсора	<Ctrl-T>

Команды обработки блоков текста

Отметить начало блока	<Ctrl-K> B
Отметить конец блока	<Ctrl-K> K
Отметить одно слово	<Ctrl-K> T
Копировать блок	<Ctrl-K> C
Удалить блок	<Ctrl-K> Y
Скрыть/ показать блок	<Ctrl-K> H
Переслать блок	<Ctrl-K> V
Прочитать блок с диска	<Ctrl-K> R
Записать блок на диск	<Ctrl-K> W

Прочие команды

Поиск	<Ctrl-Q> F
Поиск с заменой	<Ctrl-Q> A
Повторить последний поиск	<Ctrl-L>
Восстановить строку	<Ctrl-Q> L
Вызвать главное меню	<F10>