STA	NDARD C LANGUAGE						
The following notations are used: []enclosed item is optional; fnfunction; bblock; rtnreturn; ptdpointed;							
ptrpointer; exprexpres	ptrpointer; exprexpression; TRUEnon-zero value; FALSEzero value.						
char	Single character (may signed or unsigned)						
unsigned char short	Non-negative character Reduced precision integer						
int	Non-negative reduced precision integer Integer						
long	Extended precision integer						
float double	Floating point Extended precision floating point						
long double void	Extended precision floating point No type; Used for denoting: 1) no return value from fn						
ARITHMETIC	2) no argument of fn 3) general pointer base						
 If either operand is lor If either operand is do If either operand is flo All char and short ope value; otherwise it is c 	ng double the other is converted to long double. uble, the other is converted to double. at, the other is converted to float. rands are converted to ini if it can represent the original onverted to unsigned int.						
 If either operand is un If the two operands an type unsigned int, the If either operand is lor If either operand is un 	signed long the other is converted to unsigned long, re unsigned int and long and long represent all values of common type is long; otherwise it is unsigned long, g the other is converted to long.						
 If this step is reached, 	both operands must be int.						
ST							
{ local_var_decl	Block.						
<pre>statement }</pre>	The 1 oca1_var_dec1 (local variable declara- tions) is optional.						
break;	Terminates execution of for , while , do , or switch .						
continue;	Skips statement that follow in a do , for , or while ; then continues executing the loop						
do	Executes statement until expr is FALSE;						
<pre>while (expr);</pre>							
expr; for (e1;e2;e3)	Evaluates expr ; discards result. Evaluates expr e1 once; then repeatedly evaluates						
statement	e2, statement, and e3 (in that order) until e2 is FALSE: eg: for (i=1; i<=10,/; ++1) ; nole that statement will not be executed if e2 is FALSE on first evaluation; e1, e2 and e3 are onlineal: e2-1 assumed when omitted						
goto label;	Branches to statement preceded by <i>Tabe1</i> , which must be in same function as the goto. eg.: int Fn(void) { goto write:						
if (expr)	<pre>write: print("here am I");} If expr is TRUE then executes statement:</pre>						
statement	otherwise skips it.						
statement1	otherwise executes <i>statement2</i> .						
statement2	Null statement No effect eq while (t[i++]).						
return expr;	Returns from function back to caller with value of						
<pre>switch (expr) { case const1: statement</pre>	<i>expr</i> (must be an integer expression) is evaluated and then compared against integer constant exprs <i>const1</i> , <i>const2</i> ,						
<pre>break; case const2: statement</pre>	If a match is found, then the statements that follow the case (up to next break , if supplied) will be executed.						
break;	If no match is found, then the statements in the default case (if supplied) will be executed						
default: statement }							
statement	Executes Statement as long as expr is TRUE; statement might not be executed if expr is FALSE the first time it's evaluated.						
typedefic to accign a							
declaring a variable of the variable name, write the the keyword typedef. For /* define type	and particular data type. Where you'd normally write the new data type name instead. In front of everything, place example: COMPLEX */						
typedef stru { float re float im	ct al; aginary:						
<pre>} COMPLEX; /* declare var</pre>	iables with new type COMPLEX */						
COMPLEX c1,	c2, sum;						
char '	CUNSIANTS 'a' '\n'						
char string " floatf.	"ĥello"" "" F (1) 7.2f 2.e-15F -1E9f .5F						
double long double1, enumeration int	(1) 7.2 2.e-15 -1E9 .5 L (1) 7.21 2.e-151 -1E9L .5L (2) red january monday 17 -5 0./						
long int1, unsigned intu, hex integer 0,/ x,0	L (3) 2511 10,70′,/L U 17u 5U 0,7u 65535u 0,7xFF 0,7XFF 0,7xA0,70,7 ,7 0,71						
octal int X,	0,/777 0,/10,/0,/U 0,/ 573ul						
NOTES:							

Decimal point and/or scientific notation.
 Identifiers previously declared for an enumerated type; value treated as int.
 Or any int too large for normal int

TYPE QUALIFIERS

Constant object, cannot be altered by the program. External hardware or software can alter the variable, no const volatile optimalization.

OPERATORS									
OPERATOR	DESCRIPTION	EXAM	PLE ASSOCI	IATION					
++	Postincrement	ptr	+						
 r 1	Postdecrement	cour	1t μος Γ10 /1	_					
()	Function call	sqri	: (x)	~					
•	Struct member ref	chi	d.name						
-> sizeof	Ptr to struct memb	er chil	d_ptr->name of child						
++	Preincrement	++p1	r						
	Predecrement	co	ount						
*	Ptr indirection	ax *pti		⇐					
+	Unary plus	+a							
-~~~	Unary minus Bitwise NOT	-a ~0	77						
!	Logical negation	! re	ady						
(type)	Type conversion /	casting (flo	at) total/n						
/	Division	i /] i	⇒					
%	Modulus	i %	j	-					
+	Addition	valı Valı	10 /0 /	⇒					
	Left shift	bvte	2 << 4	⇒					
>>	Right shift	i >>	2						
< <=	Less than or equa	i< to i<	10,/0,/	-					
>	Greater than	i>	0,/	~					
>=	Greater than or ed	to cour	nt >= 90,/						
== !_	Equal to Not equal to	resu	ilt == 0,/ = FOF	⇒					
&	Bitwise AND	word	& 0,/77	⇒					
٨	Bitwise XOR	word	11 ^ word2	⇒					
9.9.	Bitwise OR	word	bits / %% i.10 /	_ <u>⇒</u>					
	Logical OR	j>0; i>8()./ readv						
?:	Conditional operat	or a>b	?a:b						
		lf a	greater than b then	¢					
= *= /=	Assignment opera	tors cour	-a eise b						
%= += -=		It is ea	qual to	¢					
&= ^= =		cour	it=count+2						
,	Comma operator	i=10),/ , j=0,/	→					
NOTES:									
Operators are	listed in decreasing	order of prece	dence.						
Associativity d	etermines: ⇒ grou	iping; → order	of evaluation for operand	ds with					
the same prece	edence: = c: is arouped i	right-to-left as:	a = (b = c):						
()									
	PREPROCE	SSOR ST							
STATEMENT	PREPROCE	SSOR ST	ATEMENTS						
STATEMENT	PREPROCES DESCR	SSOR ST	ATEMENTS	annear					
STATEMENT #define i	DESCR d text text in the	SSOR ST IPTION is substituted program; (eg:	ATEMENTS for <i>id</i> wherever <i>id</i> later #define BUFFERSIZE	appears 512)					
STATEMENT #define i	PREPROCES DESCR d text text in the constr al,a	SSOR ST. IPTION is substituted program; (eg: uct id(al, 2, will be rep	ATEMENTS for <i>id</i> wherever <i>id</i> later #define BUFFERSIZE a2,) is used, ar laced where they appear i	appear: 512) I gument: in text b					
STATEMENT #define i	PREPROCES DESCR d text text in the constr al,a corres	SSOR ST. IPTION is substituted program; (eg: uct id(a1, 2,will be reg ponding arg	ATEMENTS for <i>id</i> wherever <i>id</i> later #define BUFFERSIZE a2) is used, ar aced where they appear i ments of macro ac (AD) (CD) (CD) (CD) (CD) (CD)	appears 512) guments in text by all (eg					
STATEMENT #define i	PREPROCES DESCR d text text in the constr al,ai corres #def that x	SSOR ST IPTION is substituted program; (eg: uct id(a1, 2,will be rep ponding argu ine max(A,B =max(P+q,p)	ATEMENTS for <i>id</i> wherever <i>id</i> later #define BUFFERSIZE a2,) is used, ar laced where they appear i ments of macro cc (A) < (B) ?(A) : (B) s) (CA) > (B) ?(A) : (B) s) macro will be subsit	appears 512) guments in text by all (eg) means tuted fo					
STATEMENT #define i	PREPROCES DESCR d text text in the constr al,a corres #def that x x=(0	SSOR ST. APTION is substituted program: (eg: uct id(a1, 2, will be rep ponding argu- ine max(A, B ==max(p+q, r +q)>(r+s)?(p	ATEMENTS for <i>id</i> wherever <i>id</i> later #define BUFFERSIZE a2,) is used, ar laced where they appear i ments of macro ac b) ((A)>(B)?(A):(B) +5) macro will be subsi- the):(r+s)) in the program	appears 512) I guments in text by all (eg)means ituted fo m text)					
STATEMENT #define i #undef id	PREPROCES DESCR d text text in the constr al,ai corres #def that x x=((p	SSOR ST. IPTION is substituted program: (eg: uct id(a1, 2,will be rep ponding argu ine max(A, B =max(P+q, r +q)>(r+s)?(p ve definition of	ATEMENTS for id wherever id later #define BUFFERSIZE a2,) is used, ar laced where they appear im- ments of macro ac b) ((A)>(B)?(A):(B) +5) macro will be subsi- HQ):(r+5)) in the program id.	appears 512) I guments in text by all (eg)means tuted fo m text)					
STATEMENT #define i #undef id #if expr	PREPROCES DESCR d text text in the constr al,ai corres #def that x x=(0p ! Removing f cons #end	IPTION IPTION IPTION IPTION Is substituted program; (eg; uct id(al, 2,will be rep ponding argument ine max(p+q, r +q)>(r+s)?(p ve definition of tant expression tif will be proc	ATEMENTS for id wherever id later #define BUFFERSIZE a2,) is used, ar laced where they appear i ments of macro ac b) ((A)>(B)?(A):(B) +5) macro will be subsit +0):((+5)) in the program id. h expr is TRUE, stateme tessed, otherwise they will	appears 512) I guments in text by all (eg)means tuted fo m text) nts up to I not be					
statement #define i #undef id #if expr #endif	PREPROCES DESCR d text text in the constr al,ai corres #def thal x x=(0p ! Remov If cons #end	IPTION IPTION IPTION Is substituted program; (eg; uct id(al, 2,will be rep ponding arguine max(A, B =max(p+q, r +q)>(r+s)?(p ve definition of if will be proc	ATEMENTS for id wherever id later #define BUFFERSIZE a2,) is used, ar laced where they appear i ments of macro ac b) ((A)>(B)?(A):(B) +5) macro will be subsit +0):(r+5)) in the programid. h expr is TRUE, statement resesd, otherwise they will	appear: 512) I gument: in text be all (eg)means ituted fo m text) nts up tr I not be					
statement #define i #undef id #if expr #endif #if expr	PREPROCES DESCR d text text in the constr al,ai corres #def that x x=(0p ! Remov ff cons #end	ssor structure ssor substituted program: (eg: uct id(al, ,will be rep ponding arg: max(A, B max(A, B max(A, B)>(r+s)?(p ve definition of it will be proc ve definition of it will be proc ve definition of it will be proc ve definition of ve defini	ATEMENTS for id wherever id later #define BUFFERSIZE a2,) is used, ar laced where they appear i ments of macro ac b) ((A)>(B)?(A):(B) +5) macro will be subsit +0):(r+5)) in the programid. hexpr is TRUE, statements tessed, otherwise they will hexpr is TRUE statements hexpr is TRUE statemen	appears 512) I guments in text by all (eg) means tuted for m text) nts up to I not be					
#undef id #if expr #if expr #if expr 	PREPROCES DESCR d text text in the constr al,ai corres #def thal x x=(0p ! Remov if cons #end	ssor statuted ssor statuted program: (eg: uct id(al, ,will be rep ponding argumax(A) max(A) max(A) responding argumax(A) ponding argumax(A) ponding argumax(A) max(A) responding argumax(A) responding argumax(A) re	ATEMENTS for id wherever id later #define BUFFERSIZE a2,) is used, ar acced where they appear i ments of macro ac b) ((A)>(B)?(A):(B) +5) macro will be subsit +0):(r+5)) in the program id. h expr is TRUE, stateme ressed, otherwise they will h expr is TRUE, stateme ressed, otherwise these they are account of the second they are account of the second the second of the second of the second the second of the second of the second the second of the second of the second of the second of the second the second of the second of th	appears 512) I guments in text by all (eg)means tuted fo m text) nts up to l not be					
#undef id #if expr #if expr #endif #if expr #else	PREPROCES DESCR d text text in the constr al,ai corres #def that x x=(0p ! Remov if cons #end ff cons #end ff cons #els the #e	spin to tay data sport and tay to tay the tay tay the support of the tay tay the program: (eg: uct id(al, 2,will be rep ponding argumax(A, B max(A, B max(A, B)>(r+s)?(p ve definition of it f will be proce- tant expression e will be proce- lse and #er	ATEMENTS for id wherever id later #define BUFFERSIZE a2,) is used, ar acced where they appear i ments of macro ac b) ((A)>(B)?(A):(B) +5) macro will be subsit +0):(r+5)) in the program id. h expr is TRUE, stateme tessed, otherwise those dif will be processed	appear: 512) I gument: in text b all (eg) means ituted fo m text) nts up tr I not be					
#undef id #if expr #if expr #endif #if expr #else 	PREPROCES DESCR d text text in the const #def that x x=(0p Remov #const #def that x the #const #const #def that x that x the that x the that x that x the that x the that x that x that x the that x that x that x that x that x that x that x that x that x that x th	spin to tan, da. SSOR ST. APTION is substituted program: (eg: uct id(al, 2,will be rep ponding argumax(A, B max(A, B max(A, B) remax(A,	ATEMENTS for <i>id</i> wherever <i>id</i> later #define BUFFERSIZE a2,) is used, ar acced where they appear i ments of macro ac b) ((A)>(B)?(A):(B) +5) macro will be subsit +0):(r+5)) in the program <i>id</i> . h expr is TRUE, stateme tessed, otherwise those dif will be processed	appears 512) i guments in text by all (eg)means ituted fo m text) nts up te I not be nts up te between					
<pre>#undef id #define i #if expr #endif #if expr #else #endif</pre>	PREPROCES DESCR d text text in the constr al,ai corres #def thal x x=(0p ! Remov If cons #end If cons #els the #els	spin to tan, da. ssor structure is substituted program: (eg: uct id(al, 2,will be rep ponding argumax(A, B max(A, B max(A, C, A)) report of the structure max(A, B) report of the structure report of the structure st	ATEMENTS for <i>id</i> wherever <i>id</i> later #define BUFFERSIZE a2,) is used, ar acced where they appear i ments of macro ac b) ((A)>(B)?(A):(B) +5) macro will be subsit +0):(r+5)) in the program <i>id</i> . h expr is TRUE, stateme ressed, otherwise they will h expr is TRUE, stateme ressed, otherwise those dif will be processed	appears 512)) gumentbr in text by menso menso menso menso menso menso menso menso menso menso menso menso menso menso menso menso menso menso mentbr be truet by mentbr truet be truet by mentbr truet true					
STATEMENT #define i #define i #if expr #endif #else #endif #ifdef id	PREPROCES DESCR d text text in the const #def that x x=(0p ! Remov If cons #end ff cons #els the #els the #els	since the second	ATEMENTS for id wherever id later #define BUFFERSIZE a2,) is used, ar acced where they appear imments of macro acc b) ((A)>(B)?(A):(B) +5) macro will be subsit +0):(r+5)) in the program id. h expr is TRUE, statement ressed, otherwise they will h expr is TRUE, statement ressed, otherwise those dif will be processed h #define or on the c	appears 512) gument in text b lall (eg) Means tuted fo m text) Ints up to l not be					
<pre>#undef id #define i #define i #if expr #endif #if expr #endif #ifdef id </pre>	PREPROCES DESCR d text text in the const #def that x x=(0p ! Remov If cons #end ff cons #els the #els the #els the #els offerses	spin to tan, dar sport to tan, dar sport and the sport a	ATEMENTS for id wherever id later #define BUFFERSIZE a2,) is used, ar acced where they appear imments of macro cc b) ((A)>(B)?(A):(B) +5) macro will be subsit +0):(r+5)) in the program id. h expr is TRUE, statement ressed, otherwise they will h expr is TRUE, statement ressed, otherwise those id if will be processed h #define or on the c to the optional #elss	appeara 512) 15 Juneans tuted for m text) Ints up to nts up to the between orrmann occessed a like a					
<pre>statement #define i #define i #define i #if expr #endif #ifdef id #endif #ifdef id</pre>	PREPROCES DESCR d text text in the const #def that x x=(0p ! Remov If cons #end If cons #els the #els the #els offerer ! If id so offerer ! Remov ! Remov	since the task as a second sec	ATEMENTS for id wherever id later #define BUFFERSIZE a2,) is used, ar laced where they appear imments of macro acc b) ((A)>(B)?(A):(B) +5) macro will be subsit +0):(r+5)) in the program id. h expr is TRUE, statement ressed, otherwise they will h expr is TRUE, statement ressed, otherwise those dif will be processed h #define or on the c to the endif will be processed h #define or on the c to the endif will be processed h #define or on the c to the endif will be processed	appear: 512) i) gumenti n text b' gumenti all (eg pmears pmears nts up tr hot be nts up tr between ormmanu a like a					
<pre>statement #define i #define i #define i #if expr #endif #if expr #endif #ifdef ia #endif #ifdef i #ifndef i</pre>	PREPROCES DESCR d text text in the const #def that x x=(0p ! Remov if cons #end if cons #end if cons #els the #els the #els offerer #if id if if id if if id if if id if if id if if id	since the second	ATEMENTS for id wherever id later #define BUFFERSIZE a2,) is used, ar acced where they appear imments of macro acc b) ((A)>(B)?(A):(B) +5) macro will be subsit +0):(r+5)) in the program id. h expr is TRUE, statement tessed, otherwise they will h expr is TRUE, statement tessed, otherwise those dif will be processed h #define or on the c to the define or on the c fined, statements up to # tional #el se like at # iii	appear: 512) i) gumenth in text b' gumenth all (eg)means means nts up te nts up te hetween nts up te between ommanu escessed a like a send i i f.					
<pre>#undef id #define i #define i #if expr #endif #if expr #endif #ifdef id #endif #ifndef i #ifndef i</pre>	PREPROCES DESCR d text text in the constr al,ai corres #def that x x=(Q r Remoi If cons #end ff cons #end ff cons #end ff cons #els the #e ff def ff d line #e ff d ff d line #e	since the tay as a second seco	ATEMENTS for id wherever id later #define BUFFERSIZE a2,) is used, ar laced where they appear imments of macro acc b) ((A)>(B)?(A):(B) +5) macro will be subsit +0):(r+5)) in the program id. h expr is TRUE, statement ressed, otherwise those dif will be processed h #define or on the c to the endif will be processed h #define or on the c to the endif will be processed find, statements up to # thomal #else like at # in thomal #else	appearance of the second secon					
<pre>#undef id #define i #define i #if expr #endif #if expr #endif #ifdef id #endif #ifndef i #endif #include</pre>	PREPROCES DESCR d text text in the constr al,ai corres #def that x x=(Qp / Remoi If cons #end ff cons #end ff cons #end ff cons #els the #e / If id line)se #if) d If id If cons #els ff cons ff cons ff cons #els ff cons ff cons #els ff cons ff cons #els ff cons ff cons #els ff cons ff co	since the task data solution of the task data solution of the task data solution of task data solution of task data task definition of task data task definition of task data task definition of task data task definition of task data task defined (with task	ATEMENTS for id wherever id later #define BUFFERSIZE a2,) is used, ar laced where they appear imments of macro acc b) ((A)>(B)?(A):(B) +5) macro will be subsit +0):(r+5)) in the program id. h expr is TRUE, statement ressed, otherwise those dif will be processed h #define or on the c to the endif will be processed h #define or on the c to the endif will be processed find, statements up to # titional #else like at # if i 2 a program: look first	appearance in text by guments in text by guments in text by means intext of in methy in text of in methy in text of the methy in text of the methy in the second se					
<pre>#undef id #define i #define i #if expr #endif #if expr #endif #ifdef id #endif #ifndef i #endif #include</pre>	PREPROCES DESCR d text text in the constr al,ai corres #def that x x=(Qp / Remoi If cons #end / If id line)s #if) d If id If id line)s #if) d If id If sature If id If	sign to tay data so R ST. HPTION is substituted program: (eg: uct id(al, 2,will be rep ponding argumax(a, B max(a, B max(a, B max(a, C, A)) reported to tay argumax product a solution in max(a, B max(a, C, A)) reported to tay argumax product a solution in max(a, B) reported to tay argumax product a solution in max(a, B) reported to tay argumax reported to tay argumax reported to tay argumax reported to tay argumax is solution of tay argumax reported to tay argumax reported to tay argumax product a solution is defined (with is define	ATEMENTS for id wherever id later #define BUFFERSIZE a2,) is used, ar laced where they appear imments of macro acc b) ((A)>(B)?(A): (B) +5) macro will be subsit +0):((A)>(B)?(A): (B) +5) macro will be subsit +0):((A)>(B)?(A): (B) +5) macro will be subsit +0):((A)>(B)?(A): (B) +0)?(A)=(B)?(A): (B) +0)?(A)=(B)?(A)=(B)?(A)=(B)?(A)=(B)?(A)=(B)?(A)=(B)?(A)=(B)?(A)=(B)?(A)=(B)?(A)=(B)?(A)=(B)=(B)=(B)=(B)=(B)=(B)=(B)=(B)=(B)=(B	appears 512) Dimeans luted for m text by Dimeans luted for m text) Ints up to the ormmanne cessed a like a tend f .					
<pre>#undef id #define i #define i #if expr #endif #if expr #endif #ifdef id #endif #ifndef i #endif #include</pre>	PREPROCE: DESCR d text text in the corres #def that x x=(Cpp ' Remoi If cons #end ff cons #els the #e ' If id line #e '' If id line #e	since the program (eggination of the program (eg	ATEMENTS ATEMENTS for id wherever id later #define BUFFESIZE a2,) is used, ar aced where they appear i ments of macro ac b) ((A)>(B)?(A): (B)?(A): (A) to (A) (A) (A) (A) (A) (A) (A) (A) (A) to (A) (A) (A) (A) (A) (A) (A) (A) (A) to (A)	appears 512) in text by means tuted for metal in text by means tuted for means in text by means tuted for means in text by means tuted for means in text by means in text by in text by means in text by in text by means in text by in text in text in text in text in text in text in text in text in text in text in text in te					
<pre>#undef id #define i #define i #if expr #endif #if expr #endif #ifdef id #endif #ifndef i #endif #include #include</pre>	PREPROCES DESCR d text text in the constrained d text text in the corres #def that x x=(Qp ' Remo If cons #end ff cons #els the #e ' If id line) s other ff id lines ff id l	sson structure sson structure is substituted program: (eg: uct id(al, 2,will be rep ponding argu- ine max(A, B =max(p+q, r +q)>(r+s)?(q we definition of tant expression e will be proc is defined (with tatements up rise they will is defined (with tatements up rise they will as not been d processed: (op contents of f ry as source pi s contents of rd places.	ATEMENTS ATEMENTS for id wherever id later #define BUFFERSIZE a2,) is used, ar aced where they appear i ments of macro ac b) ((A)>(B)?(A): (B)?(A): (B)?(A): (B)?(A): (B)?(A): (B)?(A): (B)?(A): b) ((A)>(B)?(A): (B)?(A): (A) (A) (A)?(A) (A) (A)?(A) (A) (A) (A) (A) (appears 512) in text by means tuted for metal in text by means tuted for means in text by means tuted for means in text by means tuted for means in text by means in text by in text by means in text by in text by means in text by in text by in text by means in text by in tex					
<pre>#undef id #define i #define i #if expr #endif #if expr #endif #ifdef id #endif #ifndef i #endif #include #line n "</pre>	PREPROCES DESCR d text text in the constrained d text text in the corres #def that x x=(up) r r r r r r f corres #def that x x=(up) r r r r r r r r r r r r r	ssor structure ssor structure is substituted program: (eg: uct id(al, 2, will be rep ponding argu- ine max(A, B =max(p+q, r +q)>(r+s)?(q we definition of ve definition of it f will be proc is defined (with tant expression e will be proc is defined (with tatements up is contents of f y as source pi s contents of rid places. es subsequen rid ze subsequen	ATEMENTS ATEMENTS for id wherever id later #define BUFFERSIZE a2,) is used, ar aced where they appear i ments of macro ac b) ((A)>(B)?(A): (B)?(A): (B)?(A): (B)?(A): b)?(A)>(B)?(A): (B)?(A): (B)?(A): b)?(A)>(B)?(A): (B)?(A): b)?(A)>(B)?(A): (B)?(A): b)?(A)>(B)?(A): (B)?(A): b)?(A)>(B)?(A): (B)?(A): b)?(A)>(B)?(A): (B)?(A): b)?(A)>(A)?(A)>(A)?(A)>(A) (A)>(A)>(A)>(A)>(A)>(A)>(A)>(A)>(A)>(A)>	appears 512) means utext by means ututed for m text) in text by the not be means ints up to the ommana occessed a like a e like a e like a ff.					
<pre>#undef id #define i #define i #if expr #endif #if expr #endif #ifdef id #endif #ifndef i #endif #include #line n " NOTES:</pre>	PREPROCE: DESCR d text text in the constrained d text text in the corres #def that x x=((p ' Remo If cons #elS the #elS the #elS '' Remo ''	ssor structure ssor structure is substituted program: (eg: uct id(al, 2, will be rep ponding argu- ine max(A, B =max(p+q, r +q)>(r+s)?(p ve definition of if will be proc it ant expression if will be proc is defined (with tatements up is defined (with tatements up is defined (with tatements up is defined (with as not been d processed: (op contents of f rd places. es subsequen f 1e, beginnin	ATEMENTS ATEMENTS for id wherever id later #define BUFFERSIZE a2,) is used, ar aced where they appear i ments of macro ac b) ((A)>(B)?(A): (B)?(A): (B)?(A): (B)?(A): (B)?(A): (B)?(A): (B)?(A): b)?(A)>(B)?(A): (B)?(A): (A) to expr is TRUE, statement tessed, otherwise they will a expr is TRUE, statement tessed, otherwise they will be processed a dif will be processed a dif wil	appears 512) means luted for metally means luted for metally means nts up to the not be means nts up to the means nts up to the between ommanne essed a like a ff. in samm in					
#undef id #define i #define i #if expr #if expr #endif #ifdef id #endif #ifdef id #endif #include #include #line n " NOTES:	PREPROCE: DESCR d text text in the constrained d text text in the corres #def that x x=(up) r r r r r r f cons #def that x x=(up) r r r r r r r r r r r r r	sone structure sone structure	ATEMENTS ATEMENTS for id wherever id later #define BUFFERSIZE a2,) is used, ar aced where they appear i ments of macro ac b) ((A)>(B)?(A): (B)?(A): (B)?(A): (B)?(A): (B)?(A): (B)?(A): (B)?(A): (A) (A)?(A): (A) (A)?(A): (A) (A)?(A): (A) (A)?(A): (A) (A) (A)	appears 512) gumenth in text by means tuted for nts up to tuted for nts up to between ommann cessed a like a send i f f). in samme in samme					
#undef id #define i #define i #if expr #endif #ifdef id #endif #ifdef id #endif #include #include #line n " NOTES: Preprocessor s be continued er	PREPROCES DESCR d text text in the constrained d text text in the corres #def that x x=(up r r Remo if cons #end if cons #end if cons #end if cons #els the #c if id if	sis defined (with tate expression is defined (with tate expression if will be proce- ponding argu- ponding argu- ponding argu- ponding argu- ponding argu- ponding argu- ponding argu- proce- ponding argu- ponding argu- ponding argu- ponding argu- ponding argu- ponding argu- ponding argu- ponding argu- ponding argu- relation of the processed: (op a contents of f argu- pace argu- s contents of argu- pace argu- ponding argu- and argu- ponding argu- arg	ATEMENTS ATEMENTS for id wherever id later #define BUFFERSIZE a2,) is used, ar aced where they appear i ments of macro acio b) ((A)>(B)?(A): (B)?(A): (B)?(A): (B)?(A): b)?(A)>(B)?(A): (B)?(A): (B)?(A): b)?(A)>(B)?(A): (B)?(A): b)?(A)>(B)?(A): (B)?(A): b)?(A)>(B)?(A): (B)?(A): b)?(A)>(B)?(A): (B)?(A): b)?(A)>(B)?(A): (B)?(A): b)?(A)>(A)>(A)>(A)>(A)>(A)>(A)>(A)>(A)>(A)>	appears 512) gumenth in text by means tuted for nts up to nts up to between ommana cessed a like a tend i f f).					
#undef id #define i #define i #if expr #endif #ifdef id #endif #ifdef id #endif #include #include #line n " NOTES: Preprocessor s be continued er	PREPROCES DESCR DESCR d text text in the constr al,a corres #def that x x=(up r Remo f cons #end f cons #def that x x=(up r Remo f cons #def that x x=(up r f cons #end f cons #els the #c f d f i d line) s otherw #if d If i d line) s otherw #if d If i d line) s otherw fir corres fir stand f i d stand f i d stand f i d stand f i d stand f or f stand f or f otherw f i d stand f i d stand f or f stand f or f f or f stand f or f f or f stand s	since the second	ATEMENTS ATEMENTS for <i>id</i> wherever <i>id</i> later #define BUFFERSIZE a2,) is used, ar aced where they appear i ments of macro acio b) ((A)>(B)?(A): (B)?(A): (B)?(A): (B)?(A): b)?(A)>(B)?(A): (B)?(A): (B)?(A): b)?(A)>(B)?(A): (B)?(A): b)?(A)>(B)?(A): (B)?(A): b)?(A)>(B)?(A): (B)?(A): b)?(A)>(B)?(A): (B)?(A): b)?(A)>(A)>(A)>(A)>(A)>(A)>(A)>(A)>(A)>(A)>	appears s12) gumenti in text by means int text by means int sup to between int sup to int sup to					
#undef id #define i #define i #if expr #endif #ifdef id #endif #ifdef id #endif #include #include #line n " NOTES: Preprocessor s be continued er STORAGE CLASS	PREPROCES DESCR d text text in the constr al,a corres #def that x x=('p ' Remo' f cons #end f cons #el's the #c ' ' f cons #el's the #c ' ' f cons #el's the #c ' ' ' ' ' ' ' ' ' ' ' ' '	since the second	ATEMENTS ATEMENTS for id wherever id later whething BUFFERSIZE a2,) is used, ar aced where they appear i ments of macro cc b) ((A)>(B)?(A): (B)?(A): (B): (B): (B)?(A): (B): (B): (B): (B): (B): (B): (B): (B	appears siz) gumenti in text by means int sup to nts up to nts up to between ints up to between ints up to between ints up to between ints up to between ints and ints up to between ints and ints					
<pre>#undef id #define i #define i #if expr #endif #if expr #endif #ifdef id #endif #ifndef i #include #line n " NOTES: Preprocessor s be continued er STORAGE CLASS Static</pre>	PREPROCES DESCR d text text in the constr #def that x x=((p r Remo If cons #end If cons If cons I	sis defined (with statt expression e will be proceeded and the second statt statt expression e will be proceeded e will be proceeded and the second statt statt expression e will be proceeded e will be	ATEMENTS ATEMENTS for id wherever id later #define BUFFERSIZE a2,) is used, ar aced where they appear i ments of macro cc b) ((A)>(B)?(A): (B)?(A): (B)?(appears siz) gumentin in text bi means luted for nts up te between ints up te ints up te between ints up te ints up t					
<pre>#undef id #define i #define i #if expr #endif #if expr #endif #ifdef id #endif #ifndef i #include #line n " NOTES: Preprocessor s be continued er STORAGE cLASS Static extern</pre>	PREPROCES DESCR DESCR d text text in the constr al,a corres #def that x x=('p ' Remo' If cons #end If cons #end fi cons fi con	sin to tan, dar so on the second sec	ATEMENTS ATEMENTS for id wherever id later (#define BUFFERSIZE a2,) is used, ar aced where they appear i ments of macro cc) ((A) < (B)?(A): (B): (B)?(A): (B): (B)?(A): (B	appears siz) gumentin in text bi means lutued for nts up te between ints up te ints up te in					
<pre>#undef id #define i #define i #if expr #endif #if expr #endif #ifdef id #endif #ifdef id #endif #include #line n " NOTES: Preprocessor s be continued er STORAGE CLASS Static extern outc</pre>	PREPROCES DESCR DESCR d text text in the constrained d text text in the corres #def that x x=('p' remd' f cons #end f cons f f cons f f cons #end f f cons f f cons f f cons f f cons f f cons f f cons f f f cons f f f cons f f f cons f f f f f cons f f f f f cons f f f f f f f f f	sin to tan us. SSOR ST. APTION APTION is substituted program: (eg: uct id(al, 2will be rep ponding argu- ine max (A, B =max (p+q, r +q)>(r+s)?(p //e definition of if will be proce- if will be proce- atant expression e will be proce- e will be p	ATEMENTS ATEMENTS for id wherever id later #define BUFFERSIZE a2,) is used, ar aced where they appear i ments of macro ac b) ((A)>(B)?(A):(B):(B)?(A):(B):(B):(B):(B):(B):(B):(B):(B):(B):(B	appears siz) gumentin in text bi means luted for nts up te between ints up te ints up te between ints up te between ints up te between ints anne ints anne					
<pre>#undef id #define i #define i #if expr #endif #if expr #endif #ifdef id #endif #ifdef id #endif #include #line n " NOTES: Preprocessor s be continued er Extern STORAGE CLASS Static extern auto register</pre>	PREPROCES DESCR d text text in the constr al,a corres #def that x x=(mp f Remo f cons #def that x x=(mp f cons #def that x f cons #def that x x=(mp f cons #def f ind f cons #end f f cons #end f cons #end f cons #end f cons #end f cons #end f cons #end f cons #end f f f d f f f d f f f f m f f f f f f f f cons f f f f f cons f f f f f f f f	sin to tan, da. SOR ST . APTION is substituted program: (eg: uct id(al, 2will be rep ponding argu- ine max (A, B =max (p+q, r +q)>(r+s)?(p //e definition of if will be proce- if will be proce- if will be proce- e will be proce-	ATEMENTS ATEMENTS or id wherever id later #define BUFFERSIZE a2,) is used, ar aced where they appear i ments of macro ac b) ((A)>(B)?(A): (B)?(A): (B): (B)?(A): (B): (B)?(A): (B)?(appears siz) gumenth in text by means luted for n text) means luted for luted for lu					
<pre>#undef id #define i #define i #if expr #endif #if expr #endif #ifdef id #endif #ifdef id #endif #include #line n " NOTES: Preprocessor s be continued er Extern STORAGE CLASS Static extern auto register (omitted)</pre>	PREPROCES DESCR DESCR d text text in the conservent al,ai corres #def that x x=(mp f Remo If cons #def that x x=(mp f Remo If cons #end If cons #end f f ons #end f ons #erf d If id If id If ons #end f ons #end f ons #end f ons #end f ons #end f ons #end f ons #end f ons #end f ons #end f ons f outside f on a inside f onb ins outside f on a ons f on	sis defined (will sis defined (will sis defined (will sis defined (will statt expression e will be proceed if will be proceed if will be proceed and the proceed statt expression e will be proceed statt expression e will be proceed is defined (will stattements up is the will be proceed is defined (will stattements up is the will be proceed is contents of f or processed; (op contents of f or proceed is source pro- ry as sour	ATEMENTS ATEMENTS or id wherever id later #define BUFFERSIZE a2,) is used, ar aced where they appear i ments of macro ac b) ((A)<(B)?(A):(A):(A):(A):(A):(A):(A):(A):(A):(A):	appears 512) gumenth in text by means luted for nts up te between ommand occessed ilke a between in same in same in same coming onal. in same tered for teres a coming onal.					

NOTES:

inside fn/b

- Init at start of program execution; default is zero.

- Unital transformed in only one place w/o extern.
 Variable must be defined in only one place w/o extern.
 Variable is init each time fn/b is entered; no default value.
 Register assignment not guaranteed; restricted (implementation dependent) types can be assigned to registers.
 (addr. of) operator cannot be applied.
 Variable can be declared in only one place; initialized at start of program execution; default is zero.
 Defaults to auto.

inside fn/b

any expr

3.6

EXPRESSIONS An expression is one or more terms and zero or more operators. A term can be name (function or data object)

- constant sizeof(type)
- (expr) An expression is a constant expression if each term is a constant.
 - ARRAYS

A single dimension array **aname** of **n** elements of a specified type type and with specified initial values (optional) is declared with :

- type aname[n] = { val1, val2, ... };

If complete list of initial values is specified, **n** can be omitted. Only static or global arrays can be initialized. Char arrays can be initial by a string of chars in double quotes. Valid subscripts of the array range from O_{1} (o **n**-1. Multi dimensional arrays are declared with :

- type aname[n1][n2]... = { init_list };

Values listed in the initialization list are assigned in 'dimension order' (i.e. as if last dimension were increasing first). Nested pairs of braces can be used to change this order if desired.

- EXAMPLES:
- EXAMPLES: /* array of char */ static char hisname[] = {"John Smith"}; /* array of char ptrs */ static char *days[7] = {"Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"}; /* 3x2 array of ints */ int matrix[3][2] = { {10,/,11}, {-5,0,/}, {11,21} }; /* array of struct complex */ struct complex sensor_data[10,/0,/];
 - - POINTERS

A variable can be declared to be a pointer to a specified type by a statement of the form:

type *name;

- EXAMPLES:

- EXAMPLES: /* numptr points to floating number */ float *numptr; /* pointer to struct complex */ struct complex *cp; /* if the real part of the complex struct pointed to by cp is 0,/.0,/ */ if (cp->real == 0,/.0,/) {...} /* ptr to char; set equal to address of buf[25] (i.e. pointing to buf[25]) */ char *sptr = &buf[25]; /* store 'c' into loc ptd to by sptr */ *sptr = 'c'; /* set sptr pointing to next loc in buf */ ++sptr;

- /* set spin pointing to next not in ++sptr; /* ptr to function returning int */ int (*fptr) ();

FUNCTIONS

Functions follow this format ret_type name (arg1_dec1, arg2_dec1, ...)
{

local_var_decl statement

- return value; }

Functions can be declared extern (default) or static.

static functions can be called only from the file in which they are defined. **ret_type** is the return type for the function, and can be **void** if the function returns no value.

EXAMPLE :

3

- int length = 0,/;
 while (*s++)
 ++length;
 return length;

STRUCTURES

A structure **sname** of specified members is declared with a statement of the form struct sname

Each member declaration is a type followed by one or more member names An ${\bf n}$ -bit wide field ${\bf mname}$ is declared with a statement of the form:

- If **mname** is omitted, **n** unnamed bits are reserved; if **n** is also zero, the next field is aligned on a word boundary. **variable_list** (optional) declares variables of that structure type. If **sname** is supplied, variables can also later be declared using the format:

- /* declare complex struct */ struct complex {

A union **uname** of members occupying the same area of memory is declared with a statement of the form :

union uname

- Ł member_declaration;
- } variable_list;

Each member declaration is a type followed by one or more member names; variable_list (optional) declares variables of the particular union type. If uname is supplied, then variables can also later be declared using the format: union uname variable_list;

NOTE: unions cannot be initalized

szandi@hit.bme.hu

- ~ member_declaration; } variable_list;
- type mname:n;

- struct sname variable_list;
- EXAMPLE:
 - - float real; float imaginary;
 - }; '''' define structures */
 '* define structures */
 struct complex c1 = { 5.0, / , 0, /.0, / };
 struct complex c2, csum;
 c2 = c1; /* assign c1 to c2 */
 csum.real = c1.real + c2.real;

UNIONS

ENUM DATA TYPES			ESCAPE CHARACTERS					uble ldexp(d,n) iv_t ldiv(l1,l2)*	d*2r com			
An enumerated data with a statement of th	type en	ame with	values en	uml,e	num2 , is declared	\b ∖f	Backspace (<i>BS</i>) Form feed (<i>EF</i>)	\\ \	Backslash (χ) Doctal character value (n : 0, /-	do	uble log(d)	(. re natu
enum ename	{ enu	ml, enu	um2,	} var	iable_list;	\n	Newline (NL)	\ x h	 7) h Hexadecimal character value 	do do	uble log10,/(d) uble modf(d1,*d2)	log b rtn x
The optional varia Each enumerated va	lue is an	identifier o	ptionally fo	es of the pllowed b	particular enum type. y an equals sign and	∖r ∖t	Carriage return (CR) Horizontal tab (HT)	\"	(<i>h</i> : 0,/-9,a-f,A-F) Double quote ('')	do in	uble pow(d1,d2) t rand() *	d1 to rand
values by the compile	er, unless	the enur	is starting a n=value	construct	t is used.	\v ∖a	Vertical tab (VT) Bell (BEL)	\' \?	Single quote (') Question mark (?)	do do	uble sin(d) uble sinh(d)	hype
enum ename	e varia	able_1	ist;		tter using the format.		LIBRARY FU	NCT	IONS AND MACROS	vo	id srand(u) *	rese
EXAMPLES: /* define bo	olean	*/				Fund	ction argument types :		int n.nl.n2:	do	uble tanh(d)	hype
enum boolea /* declare v	n { f a ariab	alse, t le and	true }; inicia	lize	/alue */	uns dou	signed int u; uble d,d1,d2;		long 1,11,12; char *s,*s1,*s2;	vo	id *calloc(sul,	alloc
enum boolea if (done==t	n don rue)	e = fa] {} /*	lse; test v	alue :	*/	FIL tim	_E *f; me_t tl,tl1,tl2;		size_t su,sul,su2; fpos_t fl;	vo	su2) * id free(v) *	free l
	FOR	MATTE	D OUT	PUT		voi cha	id *v,*v1,*v2; ar and short are conver	rted to i	va_list ap; nt when passed to functions;	vo	id *memchr(v c su)	retur
printf is used to write to a file, use	write da fprin	ta to stand t f ; to 'v	dard output vrite' data	t (normal into a d	ly, your terminal). To character array, use	flo //	bat is converted to doub return code on error	le.	() return code on success	in	t memcmp($v1, v2, su$)	rtn < <,=
sprintf. The gen printf (fo	eral form	at of a prin arg1,	ntf call is : arg2,)		Cha	racter classification	TDHE	ctype.h	- vo	id	chars copy
where format is a printed. The general	characte	r string de	escribing ho	w arg1	, arg2, are to be	int	isalpha(c)	TRUE	if c is any alphabetic char if c is any control char	vo	*memcpy(v1,v2,su) id_*memmove	overl copy
%[flags][s	ize][.prec]t	type	<u>-</u>		int int	t isdigit(c) t isgraph(c)	TRUE TRUE	if c is any decimal digit 0,/-9 if c is any printable char except space	vo	(VI,V2,SU) id *memset(v_c_su)	set
- left justifi	y value (d	efault is rig	ght justify)			int int	t islower(c) t isprint(c)	TRUE TRUE	if c is any lowercase char if c is any printable char including. <i>space</i>	vo	id *realloc(v,su)	chan to it /
space precede # precede	positiv va octal valu	ilue with a	space	ue with (. /x: force display of	int	t isspace(c)	TRUE	If c is neither a control nor alphanum, char if c is one of the whitespace characters	Pro	gram contol	
decimal display le	point for f eading ze	loat value, ros	and leave	trailing z	eros for type g or G	int int	t isupper(c) t isxdigit(c)	TRUE	if c is any uppercase char if c is any hexadecimal digit 0./-9.A	vo	1d assert(<i>lexpr</i>)	then
size : is a numb number means next	er specif arg (must	ying the n be type of	minimum si f int) to prin	ze of th tf specifie	e field; * instead of es the size	int	tolower(c)	F,a conve	-f ert c to lowercase	vo in	id abort() *	COUS regis
prec: is the minim places for e and f; r	ium numb nax. num	er of digits ber of sigr	s to display nificant digit	for int is for g ; r	s; number of decimal nax. number of chars	int Data	t toupper(c) a conversion	conve	ert c to uppercase stdlib.h	- vo	(*func)(void)) * id exit(n) *	termi
for s; * instead of nu type : specifies the	Imber me	ans next a alue to be d	arg (int) to p displayed pe	rintf spea r the follo	ifies the precision wing character codes:	dou	uble atof(s)	ASCIL	to double conversion /HUGE_VAL,0,//	- ch vo	ar *getenv(s) * id longjmp	rtn pl resto
arg	dec.	oct.	hex.	HEX.	±d.dd ±d.dd e ±dd	lon dou	ng atol(s) uble	ASCII	to long conversion to double conversion; on return *s	in	(jmp_but env,n) t setjmp(jmp_buf	to rel save
unsigned short	na hu d	ho	hx	hX	default precision is	s lon	strtod(s1,*s2) ng	points ASCII	to char in s1 that terminated the scan/0,/ to long conversion, base n ; on return, * s	in	t system(s) *	exec
unsigned int long	u 1d	ο	x	x	o accimai digits	s	strtol(s1,*s2,n)	points	to char in s1 that terminated the scan /0,	Sea	arching and sorting	SALC
unsigned long float, double	lu	10	1x	1X	fe	uns s	strtoul(s1,*s2,n)	ASUI strt	ol)	vo	id *bsearch(void *key, void *base,	binar each
i same as d					Lf Le	voi	id_clearerr(f)	reset	error (incl. EOF) on file	-	int (*cmp)(void *ck. void *ce))	0,/
p a pointer, voic n store how ma	l * (impler ny charao	nentation-	defined) been displa	ayed, arg	is int *, no output	int	t fclose(f) t feof(f)	close TRUE	file /EOF/ (0,/) if end-of-file on f	vo	id qsort (void *base, sul, su2,	quick su2
hn store how ma 1n store how ma	ny charac ny charac	ters have	been displa been displa	ayed, arg ayed, arg	is short *, no output is long *, no output	int	fflush(f)	write l	buffered output to \mathbf{f} /EOF/ (0,/)		int (*cmp)(void *ck, void *ce))	com
E same as e ex g a double in f	cept disp or e form	lay E befo at, whiche	re exponer ever takes l	it instead ess spac	of e e w/o losing precision	int	fgetpos(f,*fl)	get th	e file position indicator to f1 /TRUE/(0,/) n-1 chars from f unless newline or end-of	Str	ing manipulation	conc
C a char		atring (pull	not require	ess spac	sion is given)	FIL	E *fopen(s1,s2)	file rea	ached; newline is stored in s if read /NULL/ file s1, mode s2: "w"=write, "r"=read	ch	ar *strchr(s,c) t strcmn(s1.s2)	retur
% % itself		ang (nan	notrequire	un preci	sion is given)	int	fprintf(f,s,)	"a"=a write	ppend, "b"=binary, "+"=update /NULL/ args to f using format s (see printf)	ch	ar *strcpy(s1,s2)	s1 le copy
characters in the form	nat string	not preced	ded by % a	re literall	y printed;	int	t fputc(C,T) t fputs(s,f)	write write	c to f; fth c/EOF/ s to f/EOF/ (≥0,/) su2 data items from f into v; su1 it	si	ze_t strcspn(s1,s2)	sear of s2
integer formats can o	lisplay ch	ars, short i	ints or ints.	\$;		FIL	(v,sul,su2,f)	numb	er bytes of each item /0,// (bytes read/sul f and open s1 with mode s2 (see	ch ci	ar *strerror(n)	corre
printf("%o	+ %#X	is %+0	D,/*d",	31,31	5,31+31);	int	freopen(s1,s2,f) fscanf(f,s,…)	fope read a	n) args from f using format s (see scanf)	ch	ar *strncat (s1.s2.su)	
Produces: 3 printf("%f	37 + 0 %a %#	,/X1F i .0./f	is +0,/	0,/62		int	fseek(f,1,n)	positio from	on file pointer; if n=seek_set, 1 is offse beginning; if n=seek_cur, from curren	in	t strncmp(s1,s2,su)	comp
%.2g",3.14,3. Produces: 3	14,3. .140.	14,3.14 /0./0./	4); /0./ 3.	143.	3.1	int	<pre>fsetpos(f,*fl) ftoll(f)</pre>	sets t	If n=SEEK_END, ITOM END OF THE / I RUE/ (U,/) he file position to f1 (0,/) /TRUE/ ht offset from the beginning of the file / 11 /	ch	ar *strncpy(s1,s2,su)	copy shor
	FOF	MATT	ED INF	VT		siz	e_t fwrite(v,sul,	write	su2 data items to f from v ; su1 is numbe es of each item /0.// (bytes written/su1)	cn ch	ar *strpbrk(s1,s2)	elem
scanf is used to re	ead data	from stand	dard input.	To read	data from a particular	int int	getc(f) getchar()	read r read r	next char from f /EOF/ next char from stdin /EOF/	si	ze_t strspn(s1.s2)	sear
general format of a s	canf	all is :			use sscant. me	cha	ar *gets(s)	read eof re	chars into s from stdin until newline o ached; newline not stored /NULL/	ch ch	ar *strstr(s1,s2) ar *strtok(s1,s2)	sear
where format is	a chara	cter string	g describin	ig the d	ata to be read and	int	d perror(s) t printf(s,)	write	args to stdout per format s; return args to stdout per format s; return ar of characters written 1:0 //	ı		seco to ca
an item in the format	string is :	ere ine rea	ad-in data a	are to be	stored. The format of	int int	<pre>t putc(c,f) t putchar(c)</pre>	write call f	c to f; rtn c /EOF/	Tin	e an *asctimo(*tm)	conv
*: specifies that t	t ype the field	is to be	skipped	and no	assigned (i.e., no	int int	<pre>t puts(s) t remove(s)</pre>	write	s and newline to stdout /EOF/ (≥ 0 ,/) //res the file named s (0,1) /TRUE/	cl	ock_t clock()	CPU
corresponding ptr is : <u>size</u> : a number gi	supplied i ving the r	n arg list) naximal si:	ze of the fie	eld		int voi	t rename(s1,s2) id rewind(f)	renan rewin	he the file named s1 to file s2 (0,/) /-1/ d f; calls fseek(f,0,/L,sEEK_SET)	ch do	ar *ctime(*tl) uble	conv differ
type : indicates the	e type of v	alue being	g read : hex	HFX	+d dd or +d dd e ±dd	int	t scanf(s,)	read numb	args from stdin per format s; return er of values read or EOF	st	difftime(tll,tl2) ruct tm	conv
short	hd					v01	u seudur(ť,S)	other	NULL CAUS SETVOUT(T,S,_IOFBF,BUFSIZ) Vise Calls setvbuf(f,NULL,_IONBF,BUFSIZ) Suffering mode for f the buffer is a with	st	"ymlime(*ti) ruct tm *localtime(*+1)	COOR
int unsigned int	d u	0	iix x	ил Х		S	setvbuf(f,s,n,su)	size buffer	su, n must be one of _IOFBF (fu ing), _IOLBF (line buffering). TONBF (n	ti	me_t mktime (struct tm *tptr)	alter: local
long unsigned lona	Îd Iu]0	 lx	1x		int	:	buffer write	ing) (0,/) /TRUE/ args to buffer s1 per format s2 (see	si	<pre>ze_t strftime(s1, su, s2,</pre>	write
float double					f,e,E,g,G 1f,1e,1E,1g,1G	s int	<pre>sprintf(s1,s2,) t_sscanf(s1,s2,)</pre>	prin reada	tf) args from s1 per format s2; (see scanf)	ti	struct tm *tptr) me_t time(*tl)	retur
i same as d					Lf,Le,LE,Lg,LG	+1L	*TmpTile()	return	e temporary file, open with "Wb+" mode ptr to it /NULL/ ate temporary file name: place result in -	; f —		ume ctir
p pointer (san n store number	ne as in p er of char	rintf), a s have bee	arg type is v en matcheo	/oid ** I, arg is ir	nt * , no input	int	ungetc(c.f)	s<>NI insert	c back into file f (as c wasn't read) /EOF/	Var ty	iable-type and number of pe	get i
hn store number In store number	er of char er of char	s have bee s have bee	en matcheo en matcheo	l, arg is s l, arg is lo	hort * , no input ong * , no input	int int	vfprintf(f,s,ap) vprintf(s,ap)	see v same	printf and fprintf as printf with variable argument list ap	vo	va_arg(ap, <i>type</i>) id va_end(ap)	va_s
c single chara s string of cha	icter, arg ars termin	is char[] ated by a	white-space	e charac	er, arg is char[]			va_s after t	tart must be called before and va_end he function	vo	10 va_start(ap, <i>pN</i>)	start just b
[] string of cha	ars termin in brack	ated by ar	ny char not hen followi	enclosed	between the [and]; are string terminators	1nt V	/sprintf(s1,s2,ap)	see v	printf and sprintf	_ [COMMAN	d Li
instead. NOTES:	2.2010	, u		J		Mati int	n t errno (errno.h) detect	is range	math.h,stdlib.h(*) error (ERANGE) and domain error (EDOM).	Arg pas	uments typed in on the c sed to the program through	ommai argc
A scan function retur – It reaches the term	ns when: hinating h	ull in the fa	ormat string			int dou	t abs(n) uble acos(d)	absol arcco	ute value of n sine of d /0,// $[0, /, \pi]$	ar ar	gc is a count of the number gv is an array of character	r of arg pointer
 It cannot obtain ad A conversion fails. 	ditional ir	put chara	cters to sca	in.		dou dou	uble asin(d) uble atan(d) uble atan2(d) day	arcsir arctar	le of d /0,// [- $\pi/2$,+ $\pi/2$] igent of d [- $\pi/2$,+ $\pi/2$] poont of d1 / d2 [- $\pi/2$,+ $\pi/2$]	ar	gv[u,/] points to the nai gv[argc] equal NULL po	ne of t pinter.
Any chars in format (e.g. scanf("va	string no lue=%d	t precedeo '',&iva	d by % will 1); will r	literally i natch ch	match chars on input ars "value=" on	dou	uble ceil(d) uble cos(d)	smalle cosine	est integer not less than d e of d (d in radians)	exa	mple:	annen IIS ' 9
Whitespace in forma	integer w it string m	nich will b	e read and e longest p	stored in ossible s	sequence of the zero	dou div	uble cosh(d) /_t div(n1,n2) *	hipert comp	polic cosine of d utes the quotient (.quot) and remainde	, star	ts execution of a program c	alled c
or more whitespace of EXAMPLE:	naracter	s on input.				dou	ıble exp(d)	(. ren e to th	i) of division n1/n2 ne d-th power /HUGE_VAL/		<pre>argc = 3 argv[0,/] = "che argv[1] = "mhore</pre>	ck"
sscanf("12 "%)	2Free (%c%*[/	of char ^ab]%2s	rge 21" s%d",&i	, ,&c,te	ext,&j);	dou dou	uble tabs(d) uble floor(d) uble freed(d) d2)	absol	ute value of d st integer not greater than d	To	convert number in argv[2	2], use
will return 3 and $i=3$	30,/3	c='r',t	ext="a	r"; j rer	nains unchanged.	dou lon	ושיו (מו,מב) uble frexp(d,*n) מ labs(l)	ai mo return	s x in interval (½,1), and d=x*2ⁿ ute value of 1		int main (int an { float amou	gc, nt;
20-100-16						101	.g 1003(1)	ausul			sscant (a	rgvL

d*2ⁿ computes the quotient (.quot) and remainder (.rem) of division 11/12 natural log of d /0// log base 10/ of d /0// rt x such that d1=x+d2, x in [0,1), d2 integer d1 to the d2-th power /0/,HUGE_VAL/ random number in range [0/,RAND_MAX] sine of d (ain radians) sine of **d** (**d** in radians) Sine of a (a in radians) hyperbolic sine of a square root of a /0,// reset random number generator to u tangent of d (radians) /HUGE_VAL/ hyperbolic tangent of d lation string.h,stdlib.h(*) allocate space for sul elements; each su2 bytes large and set to 0, / NULL /free block of space pointed to by v allocate su bytes and return pir to it /NULL / eturn pir in v of 1st incident of c, looking at su unsigned chars at most, or NULL if not found rtn <0, /, =0, /, >0, /if v1 is lexicographically <,= or >v2, comparing up to su unsigned chars chars copy su chars from v2 to v1 (v1, v2 should not overlap); return v1 copy su chars from v2 to v1 (v1, v2 can overlap); return v1 set su unsigned chars ptd to by v to value c; return v charped the size of block v to c;; and roturns ptr. change the size of block $\mathbf v$ to $\mathbf s \mathbf u$ and returns ptr to it /NULL/ setjmp.h,stdlib.h(*) if NDEBUG is not defined and *iexpr* is FALSE then write a diagnostic message to **stderr** and calls **abort**(): use **assert**.h header couse abnormal program termination register **func** to be called by exit (0,/) /TRUE/ terminate execution, returning exit status **n** rtn ptr to value of environment name **s** /NULL/ restore environment from **env**; causes seljmp to return **n** if **unpli**(d) **r** if n=0, / save stack environment in **env**; (0,) (see longjmp) execute s as if it were typed at terminal; returns exit status /-1/ stdlib.h binary search in array **base** (**su1** elements, each **su2** bytes large), using function **cmp** for comparison; **cmp** must return negativ if **ck<ce**, **0**, / if **ck=ce**, positiv if **ck>ce** quick sort of array **base** (**su1** elements, each **su2** bytes large), using function **cmp** for comparison; (for **cmp** see **bsearch**) string.h concatenate s2 to end of s1; rtn s1 return ptr to 1st occurrence of c in s /NULL/ compare s1 and s2; returns <0, /, 0, /, >0, / if s1 lexicographically <s2, =s2, >s2 copy s2 to s1; rtn s1 search the first s1[1] that equals any element of s2; rtn i search the first still inter equals any element of s2; th i return a pointer to string that message corrensponds to errorcode n length of s (not incl. NULL) concatenate at most su chars from s2 to end of s1; rtn s1 compare at most su chars from s1 to s2; (see compare at most su chars from s1 to s2; (see strcmp) copy at most su chars from s2 to s1; if s2 is shorter than su, null bytes are appended; th s1 searches the first s1[i] that equals any element of s2; return &s1[i] return pointer to last occurrence of c in s /NULL/ return pointer to last occurrence of c in s /NULL/ search the first s1[i] that equals none of the search the first signal duals findle of the element of s2; fit i search the first substring in s1 that matches s2 break s1 into tokens delimited by s2; from the second call s1=NULL; s2 may different from call to call; return the ptr to token or NULL time.h convert **tm struct** to string; rtn ptr to it CPU time in 1.0,//CLOCKS_PER_SEC seconds since program startup /-1/ convert time ptd to by t1 to string; rtn ptr to it difference t11-t12 in seconds convert time pointed to by t1 to Universal Time Coordinated (UTC) (formerly GMT) convert time pointed to by t1 to local time alters **tptr** to represent an equivalent encoded local time /-1/ write **tptr** to buffer **s1** per format **s2**; buffer size is **su**; rtn number of characters stored /0,// returns time & date in seconds; if t1<>NULL, time is stored in *t1; convert time returned with ctime, localtime Or gmtime /-1/ guments stdarg.h gen exit argument: **ap** must be initialized by **va_start**; the argument type must be **type** end variable argument list start variable argument list; **pV** is the parameter just before the (...) in the function prototype LINE ARGUMENTS nmand line when a program is executed are rgc and argv. f arguments +1; inters that point to each argument. e of the program executed. nter nents stored in \mathbf{argv} to other data types. For ed check, with : argv[2] = "35.79" argv[3] = NULL k" , use sscanf. For example : c, char *argv[])