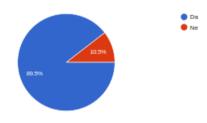
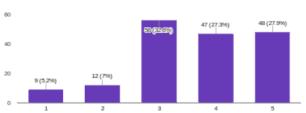
[PPI] 1. Masovne Instrukcije 2016/2017

Masovne

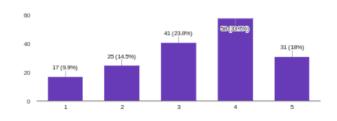




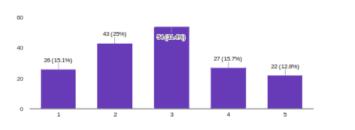
Kakve masovne želite? (172 responses)



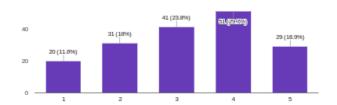
01 - Tipovi Podataka (172 responses)



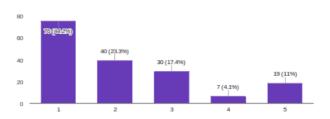
03 - Kontrolne Naredbe (172 responses)



02 - Ostali Operatori (172 responses)



04 - Polja (172 responses)



Why C Sucks!

44 Years Old
Compiled
Manual Memory Management
No run-time error checks

Why C Sucks! Rocks!

44 Years Old
Compiled
Manual Memory Management
No run-time error checks

Right Tool For The Job



C vs Python

```
    hello.c

                                                                                                                                 Raw
       #include <stdio.h>
       #define NAME_LENGHT 30
   3
       int main(void) {
  5
         char name[NAME_LENGHT + 1];
   6
         printf("What's your name? ");
         gets(name);
   8
  9
         printf("Hello %s!", name);
  10
  11
         return 0;
  12

    hello.py

                                                                                                                                 Raw
       #!/usr/bin/python
      print "What is your name? "
      name = input()
      print "Hello " + name + "!"
```

C code is more than 2x longer than Python code

Every C Program Ever

```
    basic_structure.c

                                                                                                                            Raw
      /* Include statements and preprocessor definitions */
      #include ...
      /* Function prototypes and variable declarations */
      int func(...);
      int main(int argc, char *argv[]) {
        /* The body of the main function */
  9
 10
        return 0;
 11
      /* Definitions of all other functions*/
      int func(...) {
      /* Body of some other function */
 16 }
```

Data Types

char unsigned char	1 byte
short unsigned short	2 bytes
<pre>int unsigned int</pre>	4 bytes
long unsigned long	4 bytes
float	4 bytes
double	8 bytes
long double	12 bytes

Data Types: Char, Short, Int, Long

Signed – Two's complement (complement + 1)

Char: [-128, 127]

Short: [-32768, 32767]

Int & Long: [-2147483648, 2147483647]

N-bits: [-2ⁿ⁻¹, 2ⁿ⁻¹-1]

Unsigned – ordinary binary representation

Char: [0, 255]

Short: [0, 65535]

Int & Long: [0, 4294967295]

N-bits: [0, 2ⁿ]

Data Types

Data Types/Declarations

character (1 byte)	char
integer	int
float (single precision)	float
float (double precision)	double
short (16 bit integer)	short
long (32 bit integer)	long
positive and negative	signed
only positive	unsigne

C Reference Card (ANSI)

C Reference Card (ANSI)			
Program Structure/F type fact(type_1,) type name nain() { declarations statements	functions function declaration external variable de main routine local variable declar	clarations	
} type fnc(arg1,) { declarations statements return value; }	function definition local variable declar	ations	
/* */ main(int argc, char *argv[])	comments main with args		
exit(arg)	terminate execution		
C Preprocessor			
include library file include user file replacement text replacement macro Example. #define max(A, B) undefine quoted string in replace concatenate args and rescan conditional execution	#undef name # ## ## #if, #else, #elif, #	name" : text) text)))	
is name defined, not defined?	#ifdef, #ifne		
have defined: line continuation char	defined(name	:)	
Data Types/Declarat			
Data Types/Deciarate character (1 byte) integer float (single precision) short (16 bit integer) long (32 bit integer) positive and negative only positive	char int float double short long signed unsigned		
emimeration constant constant (unchanging) value declare external variable register variable local to source file	enum const extern register static		
no value	void		
structure	struct		

sizeof object

sizeof (type name)

type name=value type name[]={value_1,...} char_name[]="string"

create name by data type size of an object (type is size_t)

Initialization initialize variable

initialize array initialize char string

size of a data type (type is size_t)

Constants

};

long (suffix)	L or 1
float (suffix)	F or f
exponential form	e
octal (prefix zero)	0
hexadecimal (prefix zero-ex)	Ox or OX
character constant (char, octal, hex)	'a', '\ooo', '\xhh
newline, cr, tab, backspace	\n, \r, \t, \b
special characters	\ \?, \', \"
string constant (ends with '\0')	"abcde"

Pointers, Arrays & Structures

declare pointer to type			type	*name
declare function returning p	pointer to	type.	type	*f()
declare pointer to function	returning	type	type	(*pf)()
generic pointer type			void	*
null pointer			NULL	
object pointed to by pointer	r		* poin	iter
address of object name			knan	i.e
array			name	: [dim]
multi-dim array		na	ne[d]	im_1][dim_2].
Structures				
struct (ag {	structure	tem	plate	
declarations	declaratio			bers

create structure	struct tag	name
member of structure from template	name.men	1ber
member of pointed to structure	pointer ->	member
Example. $(*p).x$ and $p->x$ are the	same	
single value, multiple type structure	union	
bit field with h bits	member:	h

Operators (grouped by precedence)

structure member operator structure pointer	name . member pointer->member
increment, decrement plus, minus, logical not, bitwise not indirection via pointer, address of objecast expression to type size of an object	++, +, -, !, - ect *pointer, &name (type) expr sizeof
multiply, divide, modulus (remainder)	*, /, %
add, subtract	+, -
left, right shift [bit ops]	<<, >>
comparisons	>, >=, <, <=
comparisons	, !-
bitwise and	k
bitwise exclusive or	-
bitwise or (incl)	1
logical and	kk
logical or	H
conditional expression	$expr_1$? $expr_2$: $expr$
assignment operators	+=, -=, *=,
expression evaluation separator	,
Unary operators, conditional expression	on and assignment open

Unary operators, conditional expression and assignment operators group right to left; all others group left to right.

Flow of Control

statement terminator		į ,
block delimeters		{ }
exit from switch, while,	do, for	break
next iteration of while,	do, for	continue
go to		goto label
label		label:
return value from functi	on	return expr
Flow Constructions		
if statement	if (expr) stat	
	else if (expr) else statement	
while statement	while (expr) statement	
for statement	for (expr ₁ ; expression expression)	pr ₂ ; expr ₃)
do statement	<pre>do statement while(expr);</pre>	
switch statement	switch (expr) -	
		<pre>statement1 break;</pre>
		statement ₂ break;
	default: sto	tement
	}	

ANSI Standard Libraries

<assert.h> <ctype.h> <errno.h> <float.h> inits.l
<locale.h> <nath.h> <setjnp.h> <signal.h> <stdrg.l
<stddef.h> <stdio.h> <stdlib.h> <string.h> <tine.h>

Character Class Tests <ctype.h>

	9.1
alphanumeric?	isalnum(c)
alphabetic?	isalpha(c)
control character?	iscntrl(c)
decimal digit?	isdigit(c)
printing character (not incl space)?	isgraph(c)
lower case letter?	islower(c)
printing character (incl space)?	isprint(c)
printing char except space, letter, digit?	ispunct(c)
space, formfeed, newline, cr, tab, vtab?	isspace(c)
upper case letter?	isupper(c)
hexadecimal digit?	isxdigit(c)
convert to lower case?	tolower(c)
convert to upper case?	toupper(c)

String Operations <string.h>

s,t are strings, cs,ct are constant strings

a'r ar ar me'r caire ar comann ar me'r		
length of s	strlen(s)	
copy ct to s	strcpy(s,ct)	
up to n chars	strncpy(s,ct,n)	
concatenate ct after s	strcat(s,ct)	
up to n chars	strncat(s,ct,n)	
compare cs to ct	stromp(cs,ct)	
only first n chars	strncmp(cs,ct,n)	
pointer to first c in cs	strchr(cs,c)	
pointer to last c in cs	strrchr(cs,c)	
copy n chars from ct to s	memcpy(s,ct,n)	
copy n chars from ct to s (may overlap)	memmove(s,ct,n)	
compare n chars of cs with ct	memcmp(cs,ct,n)	
pointer to first c in first n chars of cs	menchr(cs,c,n)	
put c into first n chars of cs	memset(s,c,n)	

3

Data Types: Float, Double IEEE 754

Float – [1][8][23]

$$K = BE_{(2)} + 127$$

 $Max \approx 3.4 * 10^{38}$
 $Min \approx 1.4 * 10^{-45}$
 $|\rho| \le 2^{-24} \approx 6 \times 10^{-8}$

Double – [1][11][52]

$$K = BE_{(2)} + 1023$$

 $Max \approx 1.8 * 10^{308}$
 $Min \approx 4.9 * 10^{-324}$
 $|p| \le 2^{-53} \approx 1.1 \times 10^{-16}$

Nula:
$$0 = 0...0_{(2)}$$
 $-0 = 10...0_{(2)}$
Dernormaliziran: $K = 0$
 $+\infty / -\infty$: $K = 1...1_{(2)}$ $F = 0...0_{(2)}$

Data Types: Float, Double

Prikaz realnih brojeva IEEE 754 jednostruka IEEE 754 dvostruka preciznost preciznost K = BE + 127K = BE + 1023denormalizirani broj: K = 0 denormalizirani broj: K = 0 ± ∞ ili NaN: K = 255 ± ∞ili NaN: K = 2047 najveći pozitivan broj najveći pozitivan broj $\approx 3.4 \times 10^{38}$ $\approx 1.8 \times 10^{308}$ najmanji pozitivan broj najmanji pozitivan broj ≈ 4.9 × 10⁻³²⁴ $\approx 1.4 \times 10^{-45}$

 $|\rho| \le 2^{-53} \approx 1.1 \times 10^{-16}$

 $|\rho| \le 2^{-24} \approx 6 \times 10^{-8}$

Programiranje i programsko inženjerstvo - službeni podsjetnik

verzija 2015.a

Zhak	Opis	Dekadeks vrijednost
LF	sljedeći red, novi red	10
Space	blank, praznina	32
0	znamenka nula	48
A	veliko slovo A	65
	malo slovo a	97

Pri kaz realnih brojev a

IEEE 754 jednostruka	IEEE 754 dvostruka
preciznost	preciznost
K = BE + 127	K = BE + 1023
denormalizirani broj: K = 0	denormalizirani broj: K = 0
± ∞ ili NaN : K = 255	± ∞ili NaN: K = 2047
najveći pozitiva n broj	najve di pozitivan broj
≃ 3.4 × 10 [™]	≃ 1.8 x 10 ²⁰⁶
najmanji pozjitivan broj	najmanji pozitivan broj
= 1.4 × 10 ⁴⁶	≃ 4.9 x 10 ³²⁴
$\rho \le 2^{24} \simeq 6 \times 10^{-8}$	$\rho \le 2^{63} \approx 1.1 \times 10^{16}$

double double	fabs (double x); sin (double x); cos (double x); tan (double x);	x
double	tam (double x); axin (double x); atam (double x); atam (double x); atam (double x); tamh (double x); tamh (double x); log (double x); log (double x); log (double x, double y); agrt (double x, double y); finod (double x, double y); for double x, double y); for double x, double y); for double x, for double y); for double x, for double y);	e^x $\ln x$ $\log x$ x^5 \sqrt{x} $x \mod y$ $\begin{bmatrix} x \\ x \end{bmatrix}$
at dii b		

int abs (int x);	
long labs (long x); x void exit (int status);	
<pre>void srand (unsigned int seed); int rand (void); void *malloc (size_t size);</pre>	vrača broj iz intervala [O, RANE vrača NULL, u slučaju pogreške
<pre>void free (void *block); void *realloc(void *block, size_t size);</pre>	vrača NULL u slučaju po greške

vrača broj iz intervala 10. RAND. MAX1 vrača NULL u slučaju pogreške

```
char *stropy (char *dest, const char *src);
char *strncpy(char *dest, const char *src, size_t maxlen);
char *strcat(char *dest, const char *src);
char *strncat(char *dest, const char *src, size_t maxlen);
size_t strlen(const char *s);
int stromp (const char *s1, const char *s2);
int strncmp(const char *sl, const char *s2, size_t maxlen);
char *strchr (const char *s, int c);
```

Prioritet operatora

	OPERATORI	PRIDRUŽIVANJE
_		PHILINGETVANUE
±\V±	poziv funkcije () [] referenciranje . postfiks ++	L - D
←Viši prioritet	! ~ ++ sizeof & * prefiks ++ unami + -	D - L
-	(cast)	D - L
	* / %	L - D
	binarri + -	L - D
	<< >>	L - D
z	< <- > >-	L - D
Niži prioritet→	!-	L - D
용	6	L - D
훒	^	L - D
î	1	L - D
	66	L - D
	11	L - D
	7 :	D - L
	- *- /- &- + &- ^- - <<- >>-	D - L
	,	L - D

izgled konverzijskih specifikacija kod funkcije printf

%[znak][š	i rina)[-preciznost]tip
[znak]	Objaśnjenje
ništa	desno pozicioniranje
praznina	ispisuje - predznak, a umjesto + predznaka je praznina
	lijevo pozicioniranje
+	rezultat uvijek počinje s + ili -
0	ispisuje vodeće nule
#	konvezija na alitemativan nadin: ne utječe na c s d i u ispisuje vodedu 0 za o ispisuje vodedu 0 za o ispisuje vodedu 0 za v li X uvijek ispisuje decimalnu točku za e E f uvijek ispisuje decimalnu točku i pratoče nule za g G

Data Types: Char & ASCII

Decimal	Hexadecimal	Binary	0ctal	Char	Decimal	Hexadecimal			Char	Decimal	Hexadecimal	Binary	0ctal	Char
0	0	0	0	[NULL]	48	30	110000	60	0	96	60	1100000	140	*
1	1	1	1	[START OF HEADING]	49	31	110001	61	1	97	61	1100001	141	a
2	2	10	2	[START OF TEXT]	50	32	110010	62	2	98	62	1100010	142	b
3	3	11	3	[END OF TEXT]	51	33	110011	63	3	99	63	1100011	143	c
4	4	100	4	[END OF TRANSMISSION]	52	34	110100	64	4	100	64	1100100	144	d
5	5	101	5	[ENQUIRY]	53	35	110101	65	5	101	65	1100101	145	e
6	6	110	6	[ACKNOWLEDGE]	54	36	110110	66	6	102	66	1100110	146	f
7	7	111	7	[BELL]	55	37	110111	67	7	103	67	1100111	147	g
8	8	1000	10	[BACKSPACE]	56	38	111000		8	104	68	1101000		ĥ
9	9	1001	11	[HORIZONTAL TAB]	57	39	111001	71	9	105	69	1101001	151	i .
10	A	1010	12	(LINE FEED)	58	3A	111010		:	106	6A	1101010	152	i
11	В	1011	13	[VERTICAL TAB]	59	3B	111011		;	107	6B	1101011		k
12	C	1100	14	(FORM FEED)	60	3C	111100		<	108	6C	1101100		1
13	D	1101	15	[CARRIAGE RETURN]	61	3D	111101		=	109	6D	1101101		m
14	E	1110	16	[SHIFT OUT]	62	3E	111110		>	110	6E	1101110		n
15	F	1111	17	[SHIFT IN]	63	3F	111111		?	111	6F	1101111		0
16	10	10000	20	[DATA LINK ESCAPE]	64	40	1000000		@	112	70	1110000		р
17	11		21	[DEVICE CONTROL 1]	65	41	1000001		Ā	113	71	1110001		q
18	12	10010	22	[DEVICE CONTROL 2]	66	42	1000010		В	114	72	1110010		7
19	13	10011		[DEVICE CONTROL 3]	67	43	1000011		c	115	73	1110011		5
20	14	10100	24	IDEVICE CONTROL 4)	68	44	1000100		D	116	74	1110100		t
21	15	10101		[NEGATIVE ACKNOWLEDGE]	69	45	1000101		E	117	75	1110101		u
22	16	10110	26	[SYNCHRONOUS IDLE]	70	46	1000110		F	118	76	1110110		v
23	17		27	[ENG OF TRANS, BLOCK]	71	47	1000111		G	119	77	1110111		w
24	18	11000	30	[CANCEL]	72	48	1001000		н	120	78	1111000		×
25	19		31	[END OF MEDIUM]	73	49	1001001		ï	121	79	1111001		ý
26	1A		32	[SUBSTITUTE]	74	4A	1001010		i	122	7A	1111010		ž
27	1B		33	[ESCAPE]	75	4B	1001011		ĸ	123	7B	1111011		{
28	1C	11100	34	[FILE SEPARATOR]	76	4C	1001100		Î.	124	7C	1111100		i i
29	1D		35	[GROUP SEPARATOR]	77	4D	1001101		м	125	7D	1111101		5
30	1E		36	[RECORD SEPARATOR]	78	4E	1001110		N	126	7E	11111110		~
31	1F	11111		[UNIT SEPARATOR]	79	4F	1001111		o	127	7F	1111111		[DEL]
32	20	100000		[SPACE]	80	50	1010000		P				2	[000]
33	21	100001		I	81	51	1010001		Q					
34	22	100010			82	52	1010010		Ř					
35	23	100011		#	83	53	1010011		s					
36	24	100100		Š	84	54	1010100		Ť					
37	25	100101		%	85	55	1010101		Ü					
38	26	100110		&	86	56	1010110		v					
39	27	100111		. T	87	57	1010111		w					
40	28	101000			88	58	1011000		x					
41	29	101000		1	89	59	1011001		Ŷ					
42	2A	101010		*	90	5A	1011010		ż					
43	2B	101011		+	91	5B	1011011		ī					
44	2C	101100			92	5C	1011100							
45	2D	101101			93	5D	1011101		ì					
46	2E	101110			94	5E	1011110		,					
47	2F	101111		,	95	5F	1011111							
47	2.1	TOTILI	37	/	95	31	1011111	137	_	I				

Data Types: Constant Storage

5	000000000000000000000000000000000000000	int
'5'	00110101	char (from ASCII)
5.	010000000101000000	double (8 byte)
5.f	01000001010000000	float (4 byte)
05	000000000000000000000000000000000000000	octal (4 byte)
0x5	000000000000000000000000000000000000000	hex (4 bytes)
"5"	00110101 00000000	char ary (2x1 byte)

- Poseban problem studenti imaju u predočavanju raznih "nula"
- '0' 00110000 (znakovna konstanta u 8 bita)
- '\0' 0000000 (znakovna konstanta "nul-karakter", u 8 bita)
- "0" 00110000000000 (konstantni znakovni niz, 2 puta po 8 bita)

Data Types: Constants & ASCII

Izvadak iz ASCII tablice

Znak	Opis	Dekadska vrijednost
LF	sljedeći red, novi red	10
Space	blank, praznina	32
0	znamenka nula	48
A	veliko slovo A	65
a	malo slovo a	97

verzija 2015.a PRIDRUŽIVANJE

T. -- D

D - L

Programiranje i programsko inženjerstvo - službeni podsjetnik

Pri	oritet operatora
	OPERATORI
	poziv funkcije ()

referencirani e postfiks ++ ! ~ ++ -- sizeof & prefiks ++ --

unami + binarni +

Izvadak	iz ASCII tablice	
Zhak	Opis	Dakadaka vrijednost
LF	sljedeći red, novi red	10
Space	blank, praznina	32
0	znamenka nula	48
A	veliko slovo A	65
	malo slovo a	97

Pri kaz realnih brojeva	
IEEE 754 jednostruka	IEEE 754 dvostruka
preciznost	preciznost
K = BE + 127	K = BE + 1023
denormalizirani broj: K = 0	denormalizirani broj: K = 0
± ∞ ili NaN : K = 255	± ∞ili NaN: K = 2047
najveći pozitiva n broj	najve di pozitivan broj
≃ 3.4 × 10 ³⁸	≃ 1.8 × 10 ²⁰⁴
najmanji pozjitivan broj	najmanji pozitivan broj
≃ 1.4 × 10 ⁴⁶	≃ 4.9 × 10°24
d < 2 ⁸⁴ ~ 6 ~ 10 ¹⁸	lal < 2 ⁶³ × 11 - 10 ¹⁶

double fabs (double x); double sin (double x);	x
double cos (double x);	
double tan (double x);	
double asin (double x);	
double acos (double x);	
double atan (double x);	
double sinh (double x);	
double cosh (double x);	
double tanh (double x);	
double exp (double x);	e"
double log (double x);	In x
double log10 (double x);	log x
double pow (double x, double y);	X5
double sqrt(double x);	√z
double fmod(double x, double y);	x mod y
double ceil (double x);	[x]
double floor (double x);	x
at dlib.h	

. D
L
L
L L
d
d
d
d
sto +
način
x ili X
čku za
čku i

Constants

long (suffix)
float (suffix)
exponential form
octal (prefix zero)
hexadecimal (prefix zero-ex)
character constant (char, octal, hex)
newline, cr, tab, backspace
special characters
string constant (ends with '\0')

L or 1	
F or f	
е	
0	
0x or 0X	
'a', '\ <i>ooo</i> ',	'\x <i>hh</i>
\n, \r, \t,	\b
\ \?, \',	\"
"abcde"	

include library file #include *filename>
include user file #include *filename*
replacement meet
replacement meet
Example. #define name (ear)
Example. #define nax(A, B) (A) *(B) 7 (A) *(B)

#undef name

#ifdef, #ifndef defined(name)

#if, #else, #elif, #endif

extern

sizeof (type name)

type name=value type name[]={value1,...} char name[]="string"

main(int argc, char *argv[])

undefine quoted string in replace

line continuation char Data Types/Declarations

concatenate args and rescan conditional execution

is name defined, not defined?

Data Types/Decla character (1 byte) integer float (single precision) float (double precision) short (10 bit integer) hong (32 bit integer) positive and negative only positive only positive only positive constant (unchanging) whee declare external variable register variable

create name by data type size of an object (type is size_t) size of a data type (type is size_t)

register variable local to source file no value structure

Initialization initialize variable

initialize char string

exit(arg) C Preprocessor

C Reference	Card (ANSI)	Constants	
rogram Structur oe fnc(type1,) oe name in() { declarations statements oe fnc(arg1,) {	re/Functions function declarations external variable declarations main routine local variable declarations function definition	long (suffix) float (suffix) exponential form octal (prefix zero) hexadecimal (prefix zero-ex) character constant (char, octal, hex) newline, cr, tab, backspace special characters string constant (ends with '\0')	L or 1 F or f e 0 0x or 0X 'a', '\cco', '\x\h\n', \r', \t', \b' \\\\\\\\\\\\\\\\\\\\\\\\
declarations statements return value;	local variable declarations	Pointers, Arrays & Structures declare pointer to type declare function returning pointer to type type *f()	

Pointers	Pointers, Arrays & Structures		
	ion returning points er to function retur	type *name er to type type *f() ning type type (*pf)() void * NULL	
	ed to by pointer oject name	*pointer &name	
array multi-dim ar	ray	name [dim] $name [dim_1][dim_2]$	
Structures struct declar		cture template	

declarations	declarati	on of members	
};			
create structure		struct for	name
member of structure	from template	name.mer	nber
member of pointed t		pointer ->	membe
Example. (*p).	x and p->x are:	the same	
single value, multiple	e type structure	union	

Operators (grouped by precedence)

structure member operator structure pointer	name . member pointer -> member
increment, decrement	++,
plus, minus, logical not, bitwise not	+, -, 1, -
indirection via pointer, address of obje	
cast expression to type	(type) expr
size of an object	sizeof
multiply, divide, modulus (remainder)	*, /, %
add, subtract	+, -
left, right shift [bit ops]	<<, >>
comparisons	>, >=, <, <=
comparisons	, !-
bitwise and	Ł
bitwise exclusive or	
bitwise or (incl)	1
logical and	kk
logical or	H
conditional expression	expr ₁ ? expr ₂ : expr ₃
assignment operators	+=, -=, *=,
expression evaluation separator	

Unary operators, conditional expression and assignment operators group right to left; all others group left to right

Flow	of	Contro

Flow of Contro	ol	
statement terminator block delimeters exit from switch, while, next iteration of while, go to label return value from functi	do, for	; { } break continue goto label label: return expr
Flow Constructions		
if statement	if (expr) stateles if (expr) else statemen	statement
while statement	while (expr) statement	
for statement	for (expr ₁ ; ex statement	pr ₂ ; expr ₃)
do statement	<pre>do statement while(expr);</pre>	
switch statement		: statement ₁ brea : statement ₂ brea

ANSI Standard Libraries

<assert.h> <ctype.h> <errno.h> <float.h> limits.h
<locale.h> <math.h> <setjmp.h> <sigmal.h> <stdarg.h
<stddef.h> <stdio.h> <stdlib.h> <string.h> <time.h> Character Class Tests <ctype.h>

CHARGE CHES LESSES	-coppera
alphanumeric?	isalnum(c)
alphabetic?	isalpha(c)
control character?	iscntrl(c)
decimal digit?	isdigit(c)
printing character (not incl space)?	isgraph(c)
lower case letter?	islower(c)
printing character (incl space)?	isprint(c)
printing char except space, letter, digi	t? ispunct(c)
space, formfeed, newline, cr, tab, vtab	? isspace(c)
upper case letter?	isupper(c)
hexadecimal digit?	isxdigit(c)

convert to upper case? toupper(c) String Operations <string.h>

ct to s	strcpy(s,ct)
p to n chars	strncpy(s,ct,n)
tenate ct after s	strcat(s,ct)
p to n chars	strncat(s,ct,n)
are cs to ct	stromp(cs,ct)
nly first n chars	strncmp(cs,ct,n)
er to first c in cs	strchr(cs,c)
er to last c in cs	strrchr(cs,c)
a chars from ct to s	memcpy(s,ct,n)
chars from ct to s (may overlap)	memmove(s,ct,n)
are n chars of cs with ct	memcmp(cs,ct,n)
er to first c in first n chars of cs	memchr(cs,c,n)
into first n chars of cs	memset(s.c.n)

Data Types: Type Casting

char \rightarrow short \rightarrow int int < float < double < long double

```
char c;
short int s;
                                 int n = (int)5.3
int i, iRez;
long 1;
float f:
double d;
iRez = (c + s) / 1 -
                                 n = 5
                float
                  float
       int
       long
                                 Double a = 1 / 2
          long
              float
                 float
                 double
                      double
 int
```

Operators

Operators (grouped by precedence)

	*
structure member operator structure pointer	$name.member \\ pointer->member$
increment, decrement	++,
plus, minus, logical not, bitwise not	+, -, !, ~
indirection via pointer, address of object	*pointer, &name
cast expression to type	(type) $expr$
size of an object	sizeof
multiply, divide, modulus (remainder)	*, /, %
add, subtract	+, -
left, right shift [bit ops]	<<, >>
comparisons	>, >=, <, <=
comparisons	==, !=
bitwise and	&
bitwise exclusive or	^
bitwise or (incl)	1
logical and	&&
logical or	П
conditional expression ex	pr_1 ? $expr_2$: $expr_3$
assignment operators	+=, -=, *=,
expression evaluation separator	,
Unary operators, conditional expression ators group right to left; all others group	

C Reference Card (ANSI)

Program Structure/F		float (suffix
type $fnc(type_1,)$	function declarations	exponentia
type name	external variable declarations	octal (prefi
sain() {	main routine	hexadecima
declarations	local variable declarations	character o
statements		newline, cr
}		special cha
type $fnc(arg_1,)$ {	function definition	string cons
declarations	local variable declarations	Pointer
statements		
return value;		declare poi
)		declare fun
/* */	comments	declare poi
sain(int argc, char *argv[])	main with args	generic poi
exit(arg)	terminate execution	null pointer
C Preprocessor		object poir address of

include fibrary file	#include <pre>specialities</pre>
include user file	#include "file name"
replacement text	#define name text
replacement macro	#define name(var) text
Example. #define max(A,B)	((A)>(B) ? (A) : (B))
undefine	#undef name
quoted string in replace	#
concatenate ares and rescan	***

#if, #else, #elif, #endif is name defined, not defined? name defined? line continuation char

Data Types/Declarations

character (1 byte)	char
integer	int
float (single precision)	float
float (double precision)	double
short (16 bit integer)	short
long (32 bit integer)	long
positive and negative	signed
only positive	unsigned
pointer to int, float,	*int, *float,
enumeration constant	en un
constant (unchanging) value	const
declare external variable	extern
register variable	register
local to source file	static
no value	void
structure	struct
create name by data type	typedef typename
size of an object (type is size_t)	sizeof object
size of a data type (type is size_t)	sizeof (type name)
T-141-1141	

Initialization

initialize variable	type name=value
initialize array	$type\ name[]=\{value\ 1,$
initialize char string	char name[]="string

Constants

long (suffix) float (suffix)	L or 1 F or f
exponential form	e
octal (prefix zero)	0
hexadecimal (prefix zero-ex)	Ox or OX
character constant (char, octal, hex) newline, cr, tab, backspace	'a', '\ooo', '\xhh \n, \r, \t, \b
special characters string constant (ends with '\0')	\ \?, \', \"

ers, Arrays & Structures

declare pointer to type declare function returning pointer to declare pointer to function returning		type type	(*pf)()
generic pointer type null pointer object pointed to by pointer		WULL *poin	
address of object name array multi-dim array			ie :[dim] :m ₁][dim ₂].
muru-cum array	7244	we fu	mil][um2].

Structures

struct for name create structure create structure
member of structure from template
member of pointed to structure
Example. (*pl).x and p->x are the same
single value, multiple type structure
bit field with b bits
struct
nom
membe

structure member operator structure pointer	name . member pointer-> membe
increment, decrement plus, mimus, logical not, bitwise not indirection via pointer, address of objec- cast expression to type size of an object	++, +, -, !, - *pointer, &name (type) expr sizeof
multiply, divide, modulus (remainder)	*, /, %
add, subtract	+, -
left, right shift [bit ops]	<<, >>
comparisons	>, >=, <, <=
comparisons	, !-
bitwise and	k
bitwise exclusive or	-
bitwise or (incl)	1
logical and	kk
logical or	H
conditional expression ex	pr_1 ? $expr_2$: ex
assignment operators	+=, -=, *=,
expression evaluation separator	
Unary operators, conditional expression ators group right to left; all others group	

Flow of Control

statement terminator block delimeters	4- 4	{ }
exit from switch, while,		break
next iteration of while,	do, for	continue
go to		goto label
label		label:
return value from functi	on	return expv
Flow Constructions		
if statement	if (expr) stat	ement
	else if (crpr) else statement	statement
while statement	while (expr) statement	
for statement	for (expr1; exprant)	pr_2 ; $expr_3$)
do statement	<pre>do statement while(expr);</pre>	
switch statement		statement ₁ break; statement ₂ break;

ANSI Standard Libraries

<assert.h> <ctype.h> <errno.h> <float.h> linits.l
<locale.h> <math.h> <set/pp.h> <signal.h> <stdarg.l
<stddef.h> <stdio.h> <stdlib.h> <string.h> <tine.h>

Character Class Tests <ctype.h>

alphanumeric?	isalnum(c)
alphabetic?	isalpha(c)
control character?	iscntrl(c)
decimal digit?	isdigit(c)
printing character (not incl space)?	isgraph(c)
lower case letter?	islower(c)
printing character (incl space)?	isprint(c)
printing char except space, letter, digit?	ispunct(c)
space, formfeed, newline, cr, tab, vtab?	isspace(c)
upper case letter?	isupper(c)
hexadecimal digit?	isxdigit(c
convert to lower case?	tolower(c)

String Operations <string.h>

a,t are strings, ca,ct are constant stri	1929
length of s	strlen(s)
copy ct to s	strcpy(s,ct)
up to n chars	strncpy(s,ct,n)
concatenate ct after s	strcat(s,ct)
up to n chars	strncat(s,ct,n)
compare cs to ct	stromp(cs,ct)
only first n chars	strncmp(cs,ct,n)
pointer to first c in cs	strchr(cs,c)
pointer to last c in cs	strrchr(cs,c)
copy n chars from ct to s	nencpy(s,ct,n)
copy n chars from ct to s (may overlap)	menmove(s,ct,n)
compare n chars of cs with ct	nencnp(cs,ct,n)
pointer to first c in first n chars of cs	menchr(cs,c,n)
put c into first n chars of cs	nemset(s.c.n)

Operators: Increment & Decrement

```
int i = 2, j;
j = --i;
j = i++;

i = 1
j = 1
j = 2
int i = 2, j;

j = i++;
```

Operators: Plus, Minus, Logical Not, Bit Not

```
Logical Not (!)
                                 int i = 0;
float x = 5.0f, y = 2.0f;
      + x + y \rightarrow 7.0f
      +x + +y \rightarrow 7.0f
                                 !i = 1
      +x + + y \rightarrow 7.0f
      -x+-y \rightarrow -7.0f
                                  !i = 0
      -x - -y \rightarrow -3.0f
       -x-y \rightarrow -3.0f
                                 Bit Not (~)
+x NIJE apsolutna vrijednost od x
x = -5.0f;
                                 int char = 0;
                     \rightarrow -5.0f
       +x
                                 \sim i = -128
```

Operators: Math

Operators: Bit Ops

$$2 << 2 = 8$$
 $0010 << 2 = 1000$
 $4 >> 1 = 2$
 $0100 >> 1 = 0010$

Operators: Comparators

Operators: Bitwise

```
And - &
0111 \& 1010 = 0010
      XOR - ^
0110 ^0101 = 0011
0r - |
0101 | 1100 = 1101
```

Operators: Logical

And - &&

0r - ||

DIFFERENT FROM BITWISE!!!!

Operators: Conditional (Ternary)

expr ? true : false

Operators: Asignment

_

$$n \# 2 \qquad n = n \# 2$$

Operators: Asignment

_

$$n \# 2 \qquad n = n \# 2$$

Flow of Control

Flow of Control statement terminator block delimeters exit from switch, while, do, for break next iteration of while, do, for continue goto label go to label label:return value from function return expr Flow Constructions if statement if (expr) statement else if (expr) statement else statement while statement while (expr)statementfor statement for $(expr_1; expr_2; expr_3)$ statementdo statement statementwhile (expr); switch statement switch (expr) { case $const_1$: $statement_1$ break; case const2: statement2 break; default: statement

array multi-dim array

Structures

}:

create structure

object pointed to by pointer address of object name

member of structure from template

single value, multiple type structure bit field with b bits

multiply, divide, modulus (remainder)

structure member operator structure pointer

cast expression to type size of an object

left, right shift bit ops

bitwise exclusive or bitwise or (incl)

conditional expression

assignment operators

expression evaluation separator

add subtract

comparisons

comparisons bitwise and

logical and

logical or

member of pointed to structure point

Example. (*p).x and p->x are the same

Operators (grouped by precedence)

increment, decrement
plus, mimus, logical not, bitwise not +, -, !,
indirection via pointer, address of object *pointer, &name

Unary operators, conditional expression and assignment oper

ators group right to left; all others group left to right.

> *pointer &name

name [dim] $name [dim_1][dim_2]$.

struct for name name.member pointer -> member

(type) expr

 $expr_1$? $expr_2$: $expr_3$

+=, -=, *=, ...

sizeof

*, /, %

<<. >>

C Reference C	Card (ANSI)
Program Structure/ type fnc(type1,) type name main() { declarations statements	Functions function declarations external variable declaration main routine local variable declarations
} type fnc(arg1,) { declarations statements return value; } /* */	function definition local variable declarations comments
main(int argc, char *argv[])	main with args
exit(arg)	terminate execution
C Preprocessor	
include user file replacement text replacement macro Example. Met in eax(A, B) undefine quoted string in replace concatenate args and rescan conditional execution is name defined, not defined? name defined?	#undef name # ## #if, #else, #elif, #endif #ifdef, #ifndef defined(name)
Data Types/Declara	tions
character (1 byte) integer float (single precision) float (double precision) short (16 bit integer) long (32 bit integer) positive and negative unity of the control of the	char int float double short long signed unigned unigned enum enum enum etern e
size of an object (type is size_t)	sizeof object

© 1999 Joseph H. Silverman Permissions on back. v1.3

sizeof (type name)

type name[]={value 1,...}

char name []="string

Constants	Flow of Control	
long (suffix) for f exponential form exponenti	return value from function Flow Constructions if statement if (expr) s	; { } break continue goto label label: return expr
Pointers, Arrays & Structures	else if (crp else statem	
declare pointer to tgge declare function returning pointer to tgge type *f() declare function returning tgge type *f() declare pointer to function returning tgge type (*pf()) generic pointer type unUl pointer bject pointed to by pointer **WULL WULL WULL **WULL	while statement while (expr) for statement for (expr); statement do statement do statement while(expr); switch statement witch (expr	
address of object name	switch statement switch (cap)	, .

 <assert.h>
 <ctype.h>
 <float.h>
 inits.l

 <locale.h>
 <math.h>
 <set/pp.h>
 <signal.h>
 <stdarg.l</td>

 <stddef.h>
 <stdlib.h>
 <string.h>
 <tine.h>
 Character Class Tests <ctype.h>

alphanumeric?	isalnum(c)
alphabetic?	isalpha(c)
control character?	iscntrl(c)
decimal digit?	isdigit(c)
printing character (not incl space)?	isgraph(c)
lower case letter?	islower(c)
printing character (incl space)?	isprint(c)
printing char except space, letter, digit?	ispunct(c)
space, formfeed, newline, cr. tab, vtab?	isspace(c)
upper case letter?	isupper(c)
hexadecimal digit?	isxdigit(c
convert to lower case?	tolower(c)
convert to upper case?	toupper(c)

String Operations <string.h>

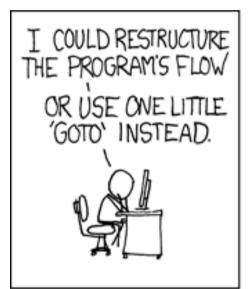
s,c are strings, cs,cc are constant strings	
length of s	strlen(s)
copy ct to s	strcpy(s,ct)
up to n chars	strncpy(s,ct,n)
concatenate ct after s	strcat(s,ct)
up to n chars	strncat(s,ct,n)
compare cs to ct	stromp(cs,ct)
only first n chars	strncmp(cs,ct,n)
pointer to first c in cs	strchr(cs,c)
pointer to last c in cs	strrchr(cs,c)
copy n chars from ct to s	mencpy(s,ct,n)
copy n chars from ct to s (may overlap)	menmove(s,ct,n)
compare n chars of cs with ct	nencnp(cs,ct,n)
pointer to first c in first n chars of cs	menchr(cs,c,n)
put c into first n chars of cs	nemset(s.c.n)

Initialization

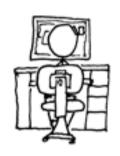
initialize variable

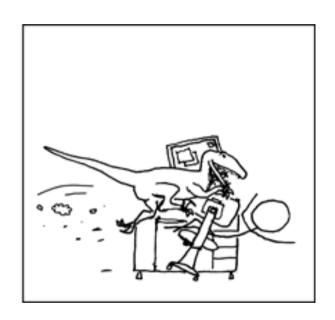
initialize char string

size of a data type (type is size_t)







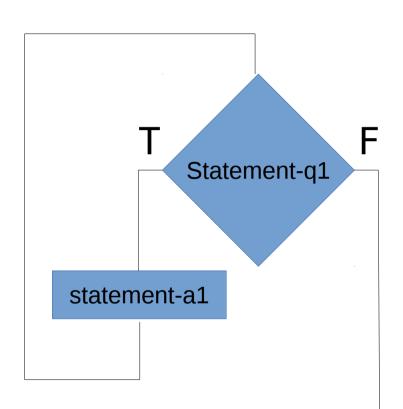


Flow of Control: If

```
F
                                     Statement-q1
if (statement-q1) {
  statement-a1;
} else if (statement-q2) {
  statement-a2;
                              statement-a1
} else {
  statement-a3;
                                            Statement-q2
                                                   statement-a3
                                     statement-a2
```

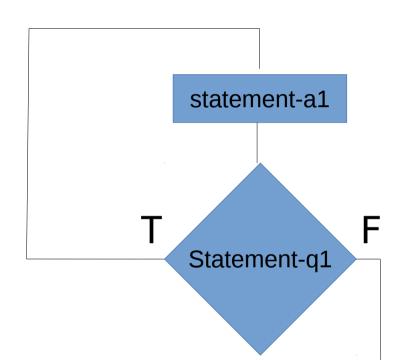
Flow of Control: While

```
while (statement-q1) {
   statement-a1;
}
```



Flow of Control: Do While

```
do {
   statement-a1;
} while (statement-q1);
```



Flow of Control: For

```
for (stat-b; stat-q; stat-e) {
  stat-a;
                                              stat-b
                                                      F
                                              stat-q
                               stat-e
                                      stat-a
```

Flow of Control: Switch

```
switch (stat-q) {
  case const-1 : {
     stat-1;
                                    Stat-q = const-1
  } case const-2 : {
     stat-2;
  } default : {
                                stat-1
     Stat-3;
                                           Stat-q = const-2
                                stat-2
                                stat-3
```

Flow of Control: Continue & Break

continue; Finishes this **one** loop iteration, ignores everything else.

> Break; Ends the loop, ignores everything else.

Flow of Control: Continue & Break

continue; Finishes this **one** loop iteration, ignores everything else.

> Break; Ends the loop, ignores everything else.

Arrays

```
int ary[size];
int ary[size] = \{1, 2, ..., size\};
int ary[size] = \{5\};
int ary[xsize][ysize][zsize];
int ary[xsize][ysize][zsize] =
  \{\{1, 2, 3, 4\},
  {5, 6, 7, 8},
  {9, 10, 11, 12}};
```

Your Mistakes

49% Syntax errors – 162 sec

31% Semantic errors – 438 sec

20% Type errors – 86 sec

[1]