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***

Prikaz realnih brojeva

IEEE 754 jednostruka preciznost	**IEEE 754 dvostruka preciznost**
K = BE + 127	K = BE + 1023
denormalizirani broj: K = 0	denormalizirani broj: K = 0
± ∞ ili NaN: K = 255	± ∞ili NaN: K = 2047
najveći pozitivan broj $\approx 3.4 \times 10^{38}$	najveći pozitivan broj $\approx 1.8 \times 10^{308}$
najmanji pozitivan broj $\approx 1.4 \times 10^{-45}$	najmanji pozitivan broj $\approx 4.9 \times 10^{-324}$
$\lvert\rho\rvert \le 2^{-24} \approx 6 \times 10^{-8}$	$\lvert\rho\rvert \le 2^{-53} \approx 1.1 \times 10^{-16}$

Prioritet operatora

	OPERATORI	*PRIDRUŽIVANJE*
←Viši prioritet	`( )`	L → D
	`! ~ ++ -- sizeof & * unarni + -`	D → L
	`(cast)`	D → L
	`* / %`	L → D
	`+ -`	L → D
	`<< >>`	L → D
	`< <= > >=`	L → D
	`== !=`	L → D
	`&`	L → D
	`^`	L → D
	`|`	L → D
	`&&`	L → D
	`||`	L → D
	`? :`	D → L
	`= *= /= %= += -= &= ^= |= <<= >>=`	D → L
Niži prioritet→	`,`	L → D

Izgled konverzijskih specifikacija kod funkcije `printf`

%[znak][širina][.preciznost]tip

[znak]	*Objašnjenje*
ništa	desno pozicioniranje
praznina	tiska - predznak, a umjesto + predznaka je praznina
-	lijevo pozicioniranje
+	rezultat uvijek počinje s + ili -
0	ispisuje vodeće nule
#	**konverzija na alternativan način:** ne utječe na c s d i u ispisuje vodeću 0 za o ispisuje vodeće 0x ili 0X za x ili X ispisuje dec. točku i kad nema decimala za e E F ispisuje prateće 0 za g G

math.h

```
int abs (int x);                                  |x|
long labs (long x);
double fabs (double x);
double sin (double 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^x
double log (double x);                            ln x
double log10 (double x);                          log x
double pow (double x, double y);                  x^y
double sqrt(double x);                            √x
double fmod(double x, double y);                  x mod y
double ceil (double x);                           ⌈x⌉
double floor(double x);                           ⌊x⌋
```

stdlib.h

```
void exit (int status);
void srand (unsigned int seed);
int rand (void);                                  vraća broj iz intervala [0, RAND_MAX]
void *malloc (size_t size);                       vraća NULL u slučaju pogreške
void free (void *block);
void *realloc(void *block, size_t size);          vraća NULL u slučaju pogreške
```

string.h

```
char *strcpy(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 strcmp(const char *s1, const char *s2);
int strncmp(const char *s1, const char *s2, size_t maxlen);
char *strchr(const char *s, int c);
```

```
char *strrchr(const char *s, int c);
char *strstr(const char *string, const char *substring);
char *strpbrk(const char *string, const char *setofcharacters);
```

ctype.h

```
int toupper(int ch);
int tolower(int ch);
int isdigit(int c);          provjerava je li znak znamenka (0-9)
int isalpha(int c);          provjerava je li znak slovo (A-Z ili a-z)
int isalnum(int c);          provjerava je li znak slovo (A-Z ili a-z) ili znamenka (0-9)
int isprint(int c);          provjerava može li se znak ispisati (0x20-0x7E)
int iscntrl(int c);          provjerava je li znak kontrolni (0x7F ili 0x00-0x1F)
int isspace(int c);          provjerava je li znak praznina
int islower(int c);          provjerava je li znak malo slovo (a-z)
int isupper(int c);          provjerava je li znak veliko slovo (A-Z)
```

stdio.h

```
int getchar(void);                    vraća učitani znak ili EOF
int putchar(int ch);                  vraća ispisani znak ili EOF (kod pogreške)
int scanf(const char *format, arg1, arg2, ...,arg n);
                                      vraća broj učitanih argumenata (0…n) ili EOF (kraj datoteke)
```

Formati za scanf: `%d,%i,%o,%u,%x,%c,%s,%e,%f,%g,%p,%[…],%[^…].`
Prefiksi: `h(za short) l(long, double) L(long double), npr. %hd, %ld, %lf, %Lf`

```
int printf(const char *format, arg1, arg2, ...,arg n);
                                      vraća broj ispisanih znakova
```

Formati za printf: `%d,%i,%o,%u,%x,%X,%c,%s,%e,%f,%g,%G,%e,%E,%p,%n.`
Zastavice (Flags) između % i formata: `-,+,razmak,0,#`

```
int puts(const char *s);              vraća EOF u slučaju pogreške
char *gets(char *string);             vraća NULL ako kao prvi znak pročita kraj datoteke (CTRL+Z (windows) ili
                                      CTRL+D(unix)) ili ako je nastupila pogreška

FILE *fopen(const char *filename, const char *mode);
```

mode: `"w","a","r","w+","a+","r+"` *Napomena: U DOS-u za neformatirane datoteke treba na kraj dodati* `b`

```
int fclose(FILE *fp);                 vraća 0 ukoliko je operacija uspjela ili EOF u slučaju pogreške
int fgetc(FILE *stream);              vraća pročitani znak ili EOF (pogreška ili kraj datoteke)
int fscanf (FILE *stream,  const char *format, arg1, arg2, ..., arg n);
                                      vraća broj učitanih argumenata ili EOF (pogreška ili kraj datoteke)
char *fgets(char *s, int n, FILE *stream);
                                      vraća NULL u slučaju pogreške ili kraja datoteke
int fputc(int c, FILE *stream);
                                      vraća ispisani znak ili EOF u slučaju pogreške
int fprintf (FILE *stream,  const char *format, arg1, arg2, ..., arg n);
                                      vraća broj ispisanih znakova ili EOF u slučaju pogreške
int fputs(char *s, FILE *stream);
                                      vraća nenegativni broj ili EOF u slučaju pogreške
size_t fread(void *ptr, size_t size, size_t n, FILE *stream);
                                      vraća broj učitanih objekata. (0..n)
size_t fwrite(void *ptr, size_t size, size_t n, FILE *stream);
                                      vraća broj ispisanih objekata. U slučaju pogreške taj je broj < n.
int fseek(FILE *stream, long offset, int whence);
                                      vraća 0 ukoliko je pozicioniranje uspjelo ili broj različit od 0 u slučaju
                                      pogreške
      whence:     SEEK_SET - pozicioniranje u odnosu na početak datoteke
                  SEEK_CUR - pozicioniranje u odnosu na trenutnu poziciju u datoteci
                  SEEK_END - pozicioniranje u odnosu na kraj datoteke
long ftell(FILE *stream);             vraća trenutnu poziciju u datoteci ili -1 u slučaju pogreške
```