POPIS FUNKCIJA IZ GLAVNIH BIBLIOTEKA TE KRATKI OPIS SVAKE OD NJIH
(Programiranje, 2. kontrolna zadaća)

math.h

int abs (int x); ----------Modul broja |x|
long labs (long x):---------- Modul broja |x|-verzija za long
double fabs (double x);----------Modul broja |x|-verzija realni br.
double sin (double x);---------- Funkcija sinus
double cos (double x);---------- Funkcija cosinus
double tan (double x);----------Funkcija tangens
double asin (double x);----------Funkcija arkussinus
double acos (double x);----------Funkcija arcuscosinus
double atan (double x);----------Funkcija arcustangens
double sinh (double x);----------Funkcija sinushiperbolni
double cosh (double x);---------- Funkcija cosinushiperbolni
double tanh (double x);----------Funkcija tangenshiperbolni
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);----------$\sqrt{x}$
double fmod (double x, double y);----------x mod y (računa ostatak dijeljena dva realna broja
double ceil (double x);----------zaokružuje argument x na najbliži veći cijeli broj
double floor (double x);----------zakružuje argument x na najbliži manji cijeli broj

stdlib.h

void exit (int status);
trenutni prekid izvođenja programa, prije prekida svi se međuspremnici prazne, datoteke se zatvaraju te se pozivaju se izlazne funkcije. Status služi da bi se pozivajući program izvijestio o eventualnoj pogrešci. Argument može biti EXIT_FAILURE ili EXIT_SUCCESS-makro imena. Ta nam funkcija služi kao izlaz u slučaju nužde- ako je nastupila neka fatalna pogreška u programu koja onemogućava daljnje izvođenje programa.

void randomize (void); ili void srand (unsigned int seed);
koriste se u skupini s ***rand*** naredbom→ inicijalizira klicu generatora slučajnih brojeva, da se ne bi uvijek generirali isti slijed slučajnih brojeva ; argument je klica pomoću koje se generira prvi broj u nizu slučajnih brojeva, svako novo pokretanje= različito polje

int rand (void); [0, RAND_MAX**]**
generiranje slučajnih brojeva; ona kao rezultat vraća slučajni broj između 0 i RAND_MAX pri čemu je broj RAND_MAX isto definiran u biblioteci stdlib, a on je najveća vrijednost koju može vratiti funkcija ***rand()***. Generator se treba inicijalizirati.

String.h

char *strcpy (char *dest, const char *src);

(*ODREDIŠTE* ← dest, *IZVOR* ← src)

preslikava sadržaj niza **IZVOR** na mjestu gdje pokazuje **ODREDIŠTE**. Preslikavanje se prekida nakon što se prenese zaključni nul- znak. Prije poziva funkcije treba alocirati dovoljan prostor za preslikani niz , jer funkcija ne provjerava je li za preslikani niz na mjestu **ODREDIŠTE** odvojeno dovoljno mjesta. Ona preslikava stručnije rečeno znakovni niz s nul znakom na koji pokazuje drugi argument , na lokaciju na koju pokazuje prvi argument. Povratna vrijednost funkcije je pokazivač na odredište preslikavanja. Iz deklaracije se očito vidi da niz koji se preslikava ostaje nepromjenjen. Međutim budući da se preslikavanjem mjenja sadržaj memorije na koju pokazuje prvi argument, neophodno je osigurati dovoljan prostor (uključujući i zaključni nul znak) u koji će se izvorni niz preslikati.

char *strncpy (char *dest, const char *src, size_t maxlen);

(**MAXDULJINA** ← maxlen)

preslikava do najviše **MAXDULJINA** znakova niza **IZVOR** na mjesto **ODREDIŠTE**. Treba uočiti ako je **MAXDULJINA** manja ili jednaka duljini niza **IZVORA**, nul znak neće biti preslikan pa niz **ODREDIŠTE** može ostati nezaključen.

char *strcat (char *dest, const char *src);

(**SPRIJEDA** ← dest, **STRAGA** ← src)

koristi se za nadovezivanje sadržaja dva znakovna niza. Djeluje tako da nadovezuje znakovni niz **STRAGA** na znakovni niz **SPRIJEDA**. Prilikom preslikavanja niza **STRAGA** preslikava se i njegov zaključni nul znak . Potrebno je isto osigurati dovoljno prostora za to preslikavanje. Funkcija kao rezultat vraća pokazivač na niz **SPRIJEDA**.

size_t strlen (const char *s);

računa duljinu znakovnog niza bez zaključnog nul znaka . Ukupno zazueće memorije je za jedan znak veće. Funkcija izračunava i vraća duljinu niza ***S.***

Size_t → rezultat tipa sizeof , cjelobrojni tip bez predznaka koji ovisi o implementaciji prevoditelja

char *strlwr (chart *s);

sva slova u nizu pretvara u mala, omogućuje konkretnu abecednu usporedbu

char *strupr (chart *s);

int strcmp (const chart *s1, const chart *s2);

(**PRVI NIZ** ← s1, **DRUGI NIZ** ← s2)

ona obavlja abecednu usporedbu sadržaja dvaju nizova znak po znak, sve dok su odgovarajući znakovi u oba niza međusobno jednaki ili dok ne naleti na nul znak jednom od nizova. Funkcija razlikuje velika i mala slova. Usporedba se obavlja prema ASCII nizu u kojem sva velika slova prethode malim slovima. Uspoređuje se **PRVI NIZ** i **DRUGI NIZ**. Rezultat usporedbe je:

<0 ako je **PRVI NIZ** manji od **DRUGOG NIZA** (svrstan po abecedi **PRVI NIZ** dolazi prije **DRUGOG NIZA**)

=0 ako je **PRVI NIZ**=**DRUGI NIZ**

>0 ako je **PRVI NIZ** veći od **DRUGOG NIZA** (svrstan po abecedi **PRVI NIZ** dolazi poslije **DRUGOG NIZA**)

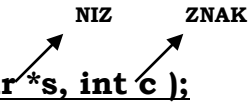
int strcmpi (const chart *s1, const chart *s2);

int stricmp (const chart *s1, const chart *s2);

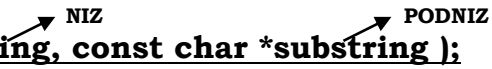
int strncmp (const chart *s1, const chart *s2, size_t maxlen);

int strncmpi (const chart *s1, const chart *s2, size_t maxlen);

int strincmp (const chart *s1, const chart *s2, size_t maxlen);



char *strchr (const char *s, int c);
funkcija u nizu **NIZ** traži prvu pojavu znaka **ZNAK**. Ako **ZNAK** nije pronađen, kao rezultat se vraća nul pokazivač. U područje pretraživanja uključen je i nul znak



char *strstr (const char *string, const char *substring);
funkcija pretražuje **NIZ** i traži prvu pojavu niza **PODNIZ** (bez njegovog zaključnog nultog znaka). Rezultat je pokazivač na početak prve pojave niza **PODNIZ** ili nul-pokazivač ako **PODNIZ** nije sadržan u **NIZU**.

ctype.h

int toupper (int ch);
ako je **ZNAK** malo slovo pretvara ga u njegovo veliko slovo. Ostali znakovi funkcije ostaju nepromjenjeni. Povratna vrijednost je kod pretvorenog znaka.

int tolower (int ch);
Ako je znak veliko slovo funkcija ga pretvara u malo slovo. Ostale znakove ostavlja nepromjenjene. Povratna funkcija je kod pretvorenog znaka.

int isdigit (int c);
vraća *true* ako je **ZNAK** naveden kao argument funkciji decimalna znamenka (0-9)

int isalpha (int c);
provjerava je li **ZNAK** slovo (A-Z ili a-z)

int isalnum (int c);
provjerava jesi li **ZNAK** slovo ili decimalna znamenka

int isprint (int c);
provjerava je li **ZNAK** znak koji se ispisuje (slovo, broj i interpunkcija), uključujući bjelinu ' '. Znak koji se može ispisati (Ox20-Ox7E)

int iscntrl (int c);
provjerava je li **ZNAK** neki kontrolni znak (ASCII kodovi 0...31 i 127)

int isspace (int c);
provjerava je li **ZNAK** praznina (bjelina, novi redak, pomak papira)

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);
učitava niz znakova sa standardnog ulaza znak po znak. Uspješno pročitan znak pretvara se u cijeli pozitivan broj, a nakon nailaska na kraj datoteke ili na pogrešku vraća EOF (end of file).

int putchar (void);
ispisuje zadani znak na standardni izlaz (zaslon računala) znak po znak.

KONTROLNI ULAZ — LISTA ARGUMENTA

int scanf (const char *format, arg1, arg2, ……,arg n);

formatirani unos podatka , KONTROLNI NIZ određuje format unesenog podatka . Uneseni podaci pridružuju se argumentima funkcije navedenih u listi argumenata. KONTROLNI NIZ sastoji se od tri grupe znakova: oznaka formata, prazno mjesto (blank), tabulator ili oznaka nove linije , ostali znakovi koji služe za povezivanje podataka s argumentima u listi argumenata. Scanf vraća broj uspješno obrađenih ulaznih polja koje povezuje s navedenim argumentima .Argumenti moraju odgovarati po broju, redoslijedu i tipu formatskim specifikacijama. S obzirom da su argumenti pokazivači, za polje se navodi njegovo ime (pokazivač na nulti član), a za obične varijable se navodi njihova adresa.

Izgled formatske specifikacije kod funkcije scanf
% [širina] [modifikator] tip [širina]
modifikator
F argument je udaljeni pokazivač
N argument je bliski pokazivač
h cjelobrojni argument je *short*
l cjelobrojni argument je *long*
tip
c jedan znak
d cijeli broj s predznakom
e, f, g broj s pomičnim zarezom
o oktalni broj
s znakovni niz (nul znak se automatski dodaje na kraj)
x heksadecimalni broj
u cijeli broj bez predznaka
[.....] znakovni niz u kojeg može biti uključena i praznina

Pojedinačne formatske specifikacije se mogu pisati neposredno jedna za drugom ili se mogu razdvojiti prazninom, Tab-om ili znakom \n.

Obično se koristi praznina. Paziti kod korištenja c-formata jer on prihvaća bilo koji unešeni znak.

int printf (const char *format, arg1, arg2, ……,arg n);

printf kao rezultat daje broj bajtova ispisanih na standardnoj izlaznoj jedinici (stdout). Argumenti mogu biti varijable, imena polja ili kompliciraniji izrazi.

Izgled formatske specifikacije kod funkcije *printf*

% [znak] [širina] [.preciznost] [modifikator] tip

[znak]

ništa	desno pozioniranje
praznina	tiska – predznak, a umjesto + predznaka je praznina
-	lijevo pozicioniranje
+	rezultat uvijek počinje + ili -
#	konverzija na alternativan način

[širina]

n	najmanje n mjesta
0n	najmanje n mjesta s tim da se lijevo stavljaju nule

[preciznost]

ništa	preciznost po definiciji	
.0	d, o, u, x	preciznost po definiciji
	e, f	bez decimalne točke
.n	najviše n znakova	

[tip]

c	znak
d	cijeli broj s predznakom
u	cijeli broj bez predznaka
o	oktalni broj bez predznaka bez vodećih nula
e	broj s pomičnim zarezom prikazan u eksponencijalnom obliku
f	broj s pomičnim zarezom
g	broj s pomičnim zarezom (e ili f oblika ovisno o vrijednosti)
s	znakovni niz
x	heksadecimalni broj bez oznake Ox ispred rezultata

int puts (const char *s);

ispisuje niz znakova na standardni izlaz. Niz znakova zadaje se kao argument funkcije. Predstavlja pokazivač na niz znakova koji se ispisuje. Oznaka za kraj niza nul znak se zamjenjuje s oznakom za novi red \n. Zauzima manje memorije i brža je od funkcije *printf ( )*

char *gets (char *string);

učitava niz znakova do unosa za oznake novi red \n. Umjesto oznake za novi red učitava oznaku za kraj niza nul znak. Prema tome funkcija gets ne može unjeti oznaku za novi red. Ako je neophodno da ' \n ' bude sastavni dio unesenog niza potrebno ga je eksplicite dodati.

IME DATOTEKE **MOD**

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

funkcija obavlja dvije operacije: definira međuspremnik i povezuje međuspremnik sa datotekom. Argument **IME DATOTEKE** predstavlja eksterno ime datoteke. Ako se podaci nalaze na disku., tada se **IME DATOTEKE** odnosi na dio diska na kojem su zapisani podaci koji se indetificiraju argumentom **IME DATOTEKE**. U slučaju datoteka, koje ne mogu djeliti svoje resurse na više logičkih dijelova, argument **IME DATOTEKE** predstavlja samo datoteku. Argument **MOD** određuje U/I operaciju. Prema definiciji funkcija vraća pokazivač na strukturu tipa **FILE** koja se još naziva i pokazivač na datoteku. Pokazivač es na datoteku deklarira kao ***FILE *fp***;. Ako se u postupku otvaranja datoteke javi pogreška , funkcija vraća null pokazivač. Greške koje nastaju otvaranjem datoteke mogu biti prouzročene otvaranjem nepostojeće datoteke za čitanjem, pokušaj otvaranjem datoteke koje su sistemski zaštićene. Greške koje nastaju otvaranjem datoteke ispisuju se preko pokazivača.

MODOVI:

«w» pisanje (ako datoteka ne postoji, stvara se; ako postoji, briše se sadržaj; nije dozvoljeno čitanje)

«a» pisanje (ako datoteka ne postoji, stvara se; ako postoji, podaci se dodaju na kraj; nije dozvoljeno čitanje)

«r» čitanje (ako datoteka ne postoji , NULL; nije dozvoljeno pisanje)

«r +» čitanje i pisanje (ako datoteka ne postoji, NULL)

«w+» čitanje i pisanje (ako datoteka ne postoji, stvara se)

«a +» čitanje i pisanje (ako datoteka ne postoji, stvara se; podaci se dodaju na kraj)

Napomena: Na DOS-u za neformatirane datoteke se dodaje **b.**

int fclose (FILE *fp);

pomoću funkcije zatvaramo datoteku. File fp je pokazivač na datoteku koji vraća funkcija *fopen()*. Upisuje preostale podatke iz međuspremnika u datoteku na disku , upisuje oznaku kraja datoteke **EOF** i prekida vezu između međuspremnika i datoteke. Zatvaranjem datoteke međuspremnik se oslobađa.Ako je uspjelo zatvaranje vraća 0 a EOF je greška.

int fgetc (FILE *stream);

funkcija omogućuje učitavanje jednog znaka. Argument predstavlja pokazivač na datoteku koji se vraća pozivom funkcije *fopen*. Funkcija vraća cijelobrojnu vrijednost pročitanog znaka i pomiče pokazivač na sljedeći znak u međuspremniku.To znači da se sadržaj datoteke čita sekvencijalno.Nakon svih učitanih podataka iz međuspremnika, u međuspremnik se učitava novi blok podataka iz datoteke. O očitvanju novog bloka u međuspremnik brinu se same funkcije. U slučaju greške funkcija vraća EOF ili kraj datoteke.Budući da EOF označava kraj datoteke, funkcija će i u slučaju redovno pročitanog kraja datoteke vratiti EOF. Budući da fgetc( ) učitava jedan znak, dakle jedan bajt, može se iskoristiti za definiranje složenih funkcija.

int fscanf (FILE *stream, const char *format, arg1,……,arg n);

funkcija za učitvanje podataka iz međuspremnika. Ona omogućuje formatirano učitavanje podataka, gdje je FILE pokazivač na datoteku koji se vraća pozivom funkcije fopen. Argumenti kontrloni niz i lista argumenata imaju isto značenje i kod funkcije scanf. Tj funkcija fscanf () je specijalni slučaj scanf funkcije gdje je datoteka iz koje se podaci učitavaju standardni ulaz. Funkcija se koristi najčešće u paru s fprint funkcijom. Funkcija vraća broj učitanih polja a greška EOF.

char *fgets (char *s, int n, FILE *stream);

s --- područje u memoriji gdje će biti smješteni podaci

n --- maksimalan broj znakova koji se može smjestiti u s

vraća pokazivač na učitani niz, Null ako je greška ili kraj datoteke.

Funkcija sprema učitane podatke na adresu na kojoj pokazuje argument funkcije s i vraća pokazivač na pročitani niz znakova.

int fputc (int c, FILE *stream);

omogućuje upis jednog znaka u međuspremnik, gdje je *stream pokazivač na datoteku kjoeg vraća funkcija fopen. Argument c zadaje se cjelobrojna vrijednost znaka koji se upisuje. Ako se u toku upisa podatka ne pojavi greška , funkcija vraća cjelobrojnu vrijednost upisanog znaka. U protivnom funkcija vraća EOF.

int fprintf (FILE *stream, const char *format, arg1, arg2, ……,arg n);

funkcija omogućuje formatirani upis podataka. Argument funkcije stream je pokazivač na datoteku, koji se vraća pozivom funkcije fopen. U slučaju da smo stream zamjenili sa stdout obje bi funkcije obavljale potpuno iste operacije, formatirani ispis podataka na standardni izlaz . Ostali argumenti funkcije imaju isto značenje kao i funkcija printf ( ). Funkcija kao rezultat vraća broj ispisanih znakova,a ako dođe do greške tada vraća EOF

int fputs (char *s, FILE *stream);

omogućuje upis niza znakova u međuspremnik, gdje s predstavlja adresu niza znakova. Ako je došlo do greške funkcija vraća EOF. U suprotnom funkcija vraća cjelobrojnu vrijednost zadnjeg upisanog znaka iz niza. U slučaju da se upisuje prazan niz funkcija vraća vrijednost nula.

KOLIKO PODRUČJE MEMORIJE ZAUZIMA

size_t fread (void *ptr, size_t size, size_t n, FILE *stream);

ptr ---------- adresa u memoriji u kojoj će se smjestiti učitani podaci
size ---------- veličina jednog objekta koji će se učitati
n ---------- broj objekata koji će se učitati pozivom funkcije; vraća broj učitanih objekata.

Ako je kraj datoteke ili pogreška, rezultat je <n.(broj učitanih blokova je manji od broja zadanih blokova). Zbog toga je kod poziva funkcije dobro provjeriti broj učitanih grupa i kraj datoteke.
Pozivom funkcije učitava se blok podataka zadanih različitim duljinama. Funkcijom može učitati bilo koji tip podatka , različitih duljina. Duljina grupe podataka koja se učitava zadaje se argumentom size. Najčešća primjena funkcije je kod učitavanja polja, polja struktura, dinamičkih struktura podataka... Najčešće se upotrebljava u paru s funkcijom fwrite ( ).

size_t fwrite (void *ptr, size_t size, size_t n, FILE *stream);

ptr ---------- adresa u memoriji s koje se zapisuju podaci
size ---------- veličina jednog objekta koji će se zapisati
n ---------- broj objekata koji će se zapisati pozivom funkcije

Vraća broj zapisanih objekata . Ako je bila greška, rezultat je <n.
Pokazivač je deklariran kao void čime je omogučen prijenos pokazivača bilo kojem tipa podatka. Osnovni blok podataka je promjenjive duljine i ovisi o tipu podatka. Duljina osnovnom bloka podatka izražava se brojem bajtova. Argument *stream je pokazivač na datoteku. Funkcija vraća cjelobrojnu vrijednost koja predstavlja broj uspješno upisanih grupa. Ako se ta vrijednost razlikuje od broja zadanih grupa došlo je do pogreške. Funkcija je pogodna za upis niza podataka različith duljina: polje, polje struktura, dinamičke strukture podataka...

ZAPIS (STRUKTURE)
→struct naziv_strukture {
tip_elementa 1 ime_elementa 1

tip_elementa n ime_elementa n
};

int seek (FILE *stream, long offset, int whence);

offset ---------- pomak u byte
whence ---------- SEEK_SET - od početka
SEEK_CUR - od trenutne pozicije
SEEK_END - od kraja datoteke
Vraća 0 ako je uspjelo, <>0 ako je pogreška
Podacima u datoteci se može pristupiti direktno, pozicioniranjem pokazivača u međuspremniku.
U slučaju pravilnom pozicioniranja pokazivača u datoteci vraća 0. a offset-broj bajtova određuje sljedeću poziciju pokazivača u datoteci koja ovisi o argumentu whence (odakle). Argument odakle određuje mjesto od kojeg će se pokazivač pomaknuti za zadani broj bajtova .

long ftell (FILE *stream);
trenutna pozicija u datoteci
vraća poziciju u datoteci ili -1 u slučaju pogreške

alloc.h

void *malloc (size_t size);
rezervira blok veličine size bajtova u memoriji računala i vraća pokazivač na taj blok. Ako blok tražene veličine nije mogao biti rezerviran vraća null pokazivač.Void= bilo koji tip pokazivača. Zauzima se memorija za zapis vrijednosti i adresa zauzete memorije pridružuje se varijabli. Veličina zauzete memorije ovisi o tipu

void free (void *block);
oslobađa blok memorije na koju pokazuje pokazivač block. Pokazivač block smije biti samo jedan od pokazivača nastalih prethodnih poziva funkcije malloc.Argument funkcije block sadrži adresu bloka memorije koji se oslobađa.

viod *realloc (void *block, size_t size);
funkcija mjenja veličinu rezerviranog bloka u memoriji. Ako se prije rezervirani blok može proširiti na veličinu size, proširuje ga u suprotnom kopira sadržaj starog bloka na novu lokaciju na kojoj ima mjesta za size byte. Ako nigdje u memoriji nema size byte slobodnog mjesta, vraća null. Ako je block NULL pokaizvač, funkcija radi kao malloc.Argument block određuje za koji podatak će se dodatno zauzeti memorija, dok se agrument size određuje koliko se dodatne memorije zahtijeva. Funkcijom se može i smanjiti zauzeće memorije.