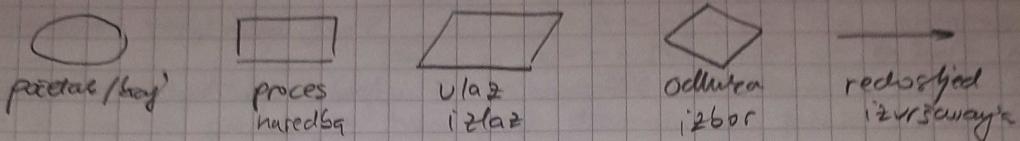


2. UVOD

- Algoritam → skup pravila koji se opisuju kako rješiti problem
- Program → opis algoritma u progr. jeziku koji je jeknoznačan

DIJAGRAM TROK



- #include <stdio.h> → predprocesorsta radnja

↳ #include → uključuje neku biblioteku
 std → standard
 i/o → input / output

- main → glavna funkcija

- return → vraca rezultat funkcije, ako ne vraca stavimo 0

- & X (u scanf) → adresa varijable; kaze nam adresu u registru na koju smo spremit vrijednost varijable

- #define PI 3.14 → predprocesorsta radnja; definicija nepromjenjivim konst.

- Compiler → prevodilac; otvara sintaktičke pogreške

- Linker → lektorijski program; povezuje potprograme biblioteke

- run-time errors → otevření by zero

- logičke greske → kvíz rozlišujeme programem

- i za predprocesorské radnji se ne stavljaj;

3. TIPOVI PODATAKA

Format

- int → integer → %d → cijeli br. #

- float → %f → realni br. #

#.3d → ako br. nije trozm. bi. dodaje 0 ispred

#.4.3f → 0000.000

- double → %.lf (scanf), %.f (printf) → dvostruka preciznost #R

- char → %c → znakovni tip / mali br.

- BIT → Binary Digit

- Prezvorite brojku: - Decimalki → Bin: dijeli u 2 i zapisuje koopacice

- Bin → Octa: grupiraju po 3 znam. S desna u lijevo i prezvorit

- Bin → heksa: — — 4 znam. — —

- Negativni br.: dvojni komplement → zapisimo 0: 1: ionda +1; 1.znm = predznak

- Raspored br.: baza moguci br. znm. -1

- Prefizi (kvalifikatori) za int:

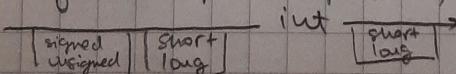
short ≤ int ≤ long

- short → smjeruje raspon

- long → povećava raspon

- Unsigned → pozitivni nule → na kraju n i %u

- Signed → pozitivni nule, i neg.; Many raspon od unsigned



- konstante: - oktalinii brojstvo \rightarrow 0 na poziciju
- heksadekadski $\rightarrow \rightarrow$ 0x - u -

- char - ASCII \rightarrow 7 bita za informacije + 1 bit za paritet $\rightarrow 2^7 = 128$ znakova

\0	\rightarrow NULL	48 - 57 \rightarrow 0 - 9
\a	\rightarrow alarm	65 - 90 \rightarrow A - Z
\n	\rightarrow novi red	97 - 122 \rightarrow a - z
\t	\rightarrow tab	127 - DEL
32	\rightarrow u	

\... upr. \101 \rightarrow sprema otkazni br.
u char
\x... upr. \x101 \rightarrow sprema heksa br.
u char

'A' - sprema redni br u ASCII

/. c \rightarrow des spremano/citamo char kao znak
/. d \rightarrow -11- mali broj

- string \rightarrow na kraju obavezno '\0'

\rightarrow \. s

\rightarrow " " = konstantan znakovni niz

- definiraju varijable: char ime [duzina+1] \rightarrow polje znakova

Decimalne br. u binarnom

- dekadski \rightarrow bin: množimo ujesta iz decimalnog zareda s 2 i prim. znim. kaj se uklazi ispred zarezu zapisujemo u binarni zapis; radimo da se učinimo 0.0 da je zadovoljena preciznost -n
- bin \rightarrow dekadski: množimo s 10 \rightarrow s lijeva na desno ide n

Realni br. u binarnom

- Jednostavna precizost:

- normalizirani br. \rightarrow omogućava priček jasno veličinu i malih br.

$$101.11 = \underbrace{1.0111}_{mantisa} \cdot 2^2 \rightarrow \text{binarni eksponent}$$

- IEEE 754 \rightarrow standard koji definira način pohrane realnih br.
- float \rightarrow jednostavna preciznost \rightarrow 32 bita
 - 1 bit za predznak, 8 za karakteristiku, 23 za mantisu bez skrivenebita
 - karakteristika = 127 + binarni eksponent (BE)

- Postupak:
1. Realni dec. br. prikazujemo u obliku realnog bin. br.
 2. Odrediti predznak
 3. Normalizirati binarni br.
 4. Izračunati karakteristiku i prenijeti u binarni br.
 5. Izračunati vodeći jedinica iz mantise (skiveni bit)
 6. Prepisati predznak, karakteristiku i mantisu bez skrivenebita u registar

Posebni slučevi:

1. k=0 i m=0 \rightarrow prikazujemo 0

2. k=0 m≠0 \rightarrow denormalizirani br.

\rightarrow Smatra se da je vodeći bit mantiše 0

\rightarrow vrijednost eksponenta je fiksirana na -126

3. k=255 m=0 \rightarrow +∞, -∞ \rightarrow osim 0 1. bitu

4. k=255 m≠0 \rightarrow NaN \rightarrow Not a number \rightarrow pogreska

- veće br. 2brog veličog br. decimala je moguće pribrojati samo približno
- veća preciznost \rightarrow moguće pribrojati više brojeva
 \rightarrow veličina pogreške pribaze je manja

- Apsolutna pogreška $\alpha = \frac{\text{br. pohranjen}}{\text{prihvatu}} - \text{stvarni br.}$

- Relativna pogreška $\rho = \frac{\alpha}{\text{stvarni br.}}$

- Numeričke pogreške \rightarrow brojni i beskonačnim decimalnim
 \rightarrow broj je prenosi u treću preniju decimala za pribaz

- Dvostruka preciznost: \rightarrow double; 64 bita

- 1 bit za predznak, 11 za karakterističnu, 52 za mantisu
- $E = BE + 1023$

Posebni slučajevi:

- $E = 0, m = 0 \rightarrow 0$
- $E = 0, m \neq 0 \rightarrow$ de-normaliziran br.
- $E = 2047, m = 0 \rightarrow \pm\infty \rightarrow$ ovise o 0 bitu za predznak
- $E = 2047, m \neq 0 \rightarrow \text{NaN}$

- float \leq double \leq long double
 \hookrightarrow f na kraju 0 \hookrightarrow L na kraju

- Veličine koje se podizajujuju za GCC:

- char = 1 byte
- short = 2
- int = 4
- long = 4
- float = 4
- double = 8
- long double = 12

Jednoznačna prec.

$$K \approx 6 \cdot 10^{-8} \cdot |x|$$

\hookrightarrow br. sig.
prihvatljivo

$$\rho \approx 6 \cdot 10^{-8}$$

Pogreške dvostrukice:

$$\alpha \approx 1.1 \cdot 10^{-16} \cdot |x|$$

$$\rho \approx 1.1 \cdot 10^{-16}$$

- Prioriteti tipova podataka:

1. long double
2. double
3. float
4. long
5. short, char \rightarrow int

- sizeof(var) \rightarrow vraca veličinu prostora potrebnu za pohranu var

Matematička logika

- Atomni/osnovni sud \rightarrow utvrdjeno nepravilno zaključivanjem

- Složeni sud: odorni sud + log. operator + zagrada

T negacija
V disjunkcija \rightarrow LI

\wedge konjunkcija \rightarrow I

\oplus ekskluzivna disjunkcija $\rightarrow XLI / XOR \rightarrow \begin{cases} 0, & \text{ako su jednaki} \\ 1, & \text{ako su razliciti} \end{cases}$

- u C ne postoji bool varijabla, zato moramo sami definirati:

```
#define TRUE 1
#define FALSE 0
```

- && \rightarrow logičko I
- || \rightarrow logičko LI
- ! \rightarrow logičko NE

TEHNIKA DVOJNOG KOMPONENTA:

- sve brojeve (i pozitivne i negativne) pretvaraju u binarno
 - ako je br. pozitivan → gotovi smo
 - " " negativan → inverz (zamjena 0:1) i fiks +1

4. Ostali operatori:

- (tip) → pretvara tip podataka upr.: i je int ; float) i --- → i je sada float
- Unarni + i - → normalno izlazuje i odn zimanje
- Operatori povećavaju i smanjuju za 1
 - ↳ prefiksmi: ++ i -- i
 - ↳ postfiksmi: i++ i--
- prefiksmi → prvo uveća / umanji vrijednost pa onda iskoristi
- postfiksmi → prvo iskoristi vrijednost varijable pa onda uveća / umanji

Binarni operatori:

* , / , % , + , - , < , > , <= , >= , == , !=

<< pomak bitova u lijevo n=n>>2 → bitovi za 2 mesta u lijevo
 >> pomak bitova u desno

& → logički / po bitovima
 ^ → isključivi ili po bitovima → XOR
 | → vključivi ili po bitovima → OR

&& → logički / → ako su oba br ≠ 0 → 1, ako je 1 ili 0 = 0 → 0
 || → logički ili

~ → unarni NOT, odnosi se na operand s desna

b	n _b
0	1
1	0

Teruarui operatori:

- uvjetno pitanjivače: uvjet ? izraz 1 : izraz 2
 → ako je uvjet true → izraz 1 else → izraz 2
- višestruko pitanjivoče → pitanjivoče vrijedost 5 desna varijabli s cijevom
 $a=b=c=0 \rightarrow a=(b=(c=0))$
- Strukuri izrazi pitanjivoče -= , += , *= , /= , %= , &= , ^= , |= , <<= , >>= $x=x+5 \rightarrow x+=5$
- Naredbe možemo sklopiti za razom, koristimo tamo gdje je obično dopuštena samo 1 naredba upr. $x=7, y=3;$

5. Kontrolne naredbe

IF → dvostrana selekcija

```

if (uvjet) {
    niz naredbi
}
else {
    niz naredbi
}
  
```

} else if

- PAZI! → else pripada najblizjem
if-u koji verna svoj else

WHILE → ispitivaje uvjeta ponavljaju na početku

```

while (uvjet) {
    niz naredbi
}
  
```

DO WHILE → ispitivaje uvjeta ponavljaju na kraju

```

do {
    niz naredbi
}
while (uvjet);
    do je (uvjet)
  
```

→ naredbe se
izvrsi barem 1.
⇒ ako uvjet nije
zadovoljen

FOR → petja s početnim br. ponavljaju

```

for (i = početak; i <= kraj; i++) {
    niz naredbi;
}
  
```

→ korak povećavajuć

- % g → formatска specifikacija za ispis realnog broja u "čvorstvu" ili "standardu"
- break → prekida izvodenje najbliže vanjske programske petje

- continue → oduzeti se na najbližu vanjsku petju

↳ while i do-while → usmjerava na ispitivanje uvjeta

↳ for → usmjerava na 3. izraz u for petji tj. povećanje/smanjenje i,
i onda na ispitivanje uvjeta 2. izraza

GO TO

goto Oznaka-naredbe;

Oznaka-naredbe:

programski-odejedale;

SWITCH → može se upotrijebiti umjesto višestruke if selekcije

Switch (cijelobrojni izraz) {

case konstanta-1 : naredbe 1;
case konst. - 2 : naredbe 2;

default : naredbe;

}

→ ako se unutar naredbi u case-u ne naredi break; raditi se spletci case

6. Podatkovna struktura polje

- L-value \rightarrow vrijednost/variabla koja nas upućuje na adresu registra za pohranu
- Polje \rightarrow složeni tip podataka (data aggregate) \rightarrow ima više članova
 \rightarrow Sistem članu pristupamo помоћу indexa

$y =$	<table border="1"><tr><td>5</td><td>10</td><td>15</td><td>20</td><td>25</td></tr><tr><td>Index:</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td></tr></table>	5	10	15	20	25	Index:	0	1	2	3	4
5	10	15	20	25								
Index:	0	1	2	3	4							

$$y[0] = 5 \quad y[3] = 20$$

- $y[0], y[1], \dots$ su L-values

- Moramo odrediti: ime, tip podataka unutar polja, broj članova polja

Opcija 1:

define MAX 20

:

float niz[MAX]; \rightarrow Polje 20 realnih br.

Opcija 2:

= float niz[20];

- članovima polja pristupamo помоћу indexa \rightarrow mora biti nenegativan br.
(znači ne možemo kao u Pythonu sa -1 povratiti zadnji član)

- članovi polja susjedni po indexu su susjedni i u registru

- možemo odmah dodijeliti vrijednosti članovima:

int Znm[10] = {1, 2, 3, 4, 5, 6, 7};
Znm =

1	2	3	4	5	6	7	0	0	0
index = 0	1	2	3	4	5	6	7	8	9

\Rightarrow članovi za koji nije uvedena postavka vrijednost se postavljaju na 0

- najlakše postavljaće svih čl. na 0: int niz[1000] = {0};

String \rightarrow konstantu pišemo unutar " "

- u C-u postoji tip string (variabla znakova) nego koristimo char polje

MORAMO viacinu \0 u veličini polja!

Ivan \rightarrow 4+1 \Rightarrow char niz[5] = {'I', 'v', 'a', 'n', '\0'}

- \0 \rightarrow shodno bilo koji od njih je 'kraj' stringa, mora li se nastavili ispisivati dalje vrijednosti sljedećih registara

IU char niz[] = "ivan"; \rightarrow Preostalo sam odredio dužinu, dodao '\0'

- gets(niz); \rightarrow pročita niz znakova, spremini ga u niz, dodala '\0' na kraj

Višedimenzionalni polje

- jednodimenzionalno polje \rightarrow vektor
- dvije dimenzije \rightarrow matrica, tablica

int mat[red][stupac]

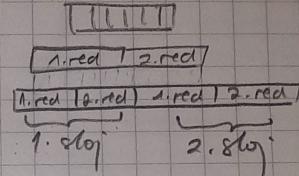
int b[2][4]

	0	1	2	3
0	b[0][0]	-	-	b[0][3]
1	-	-	b[1][2]	-

- višedimenzionalna \rightarrow neima organizaciju
 \hookrightarrow više slojeva

- Način pohrane u registru:

- jednodimenzionalno \rightarrow elem je elemom
- dvodimenzionalno \rightarrow red je redom
- tridimenzionalno \rightarrow sloj je slojem



- Vodićemo je da definisuje konstante konstante:

define MAXRED 10

define MAXSTUP 20

int mat[MAXRED][MAXSTUP];

- \Rightarrow kod višedimenzionalnog polja dimenzije MORAJU biti zadane
- Zadajemo pozitivne vrijednosti kao listu unutar liste

sizeof

int x[3][2]; veličina sačetnika br polja

sizeof(x) \rightarrow 24 \rightarrow 6 * sizeof(int)

sizeof(x[1][1]) \rightarrow 4 \rightarrow sizeof(int)

\hookrightarrow int = 4 octeta