OSNOVNI POJMOVI

Algoritam - procedura kojom se neki problem rešava

Mašinska instrukcija - to je instrukcija koju procesor izvršava (mašinski jezik (kod), kao što je Asembler (Assembly language)

Operator - su instrukcije kojom se vrši operacija, (+ , - , / , *)
Operandi - su podaci nad kojima se vrši operacija (1 i 1 , su operandi, a operator + definiše operaciju nad tim dva podatka; i time 1 + 1)

Interpreteri - je kompjuterski program (programski jezici) koji izvršavaju naredbe kako idu do nje; naredbe se izvršavaju u real time (programski jezici kao što su Python, Mathlab, Perl)

Kompajleri - je kompjuterski program koji transformiše kod jednog programskog jezika u drugi programski jezik. Kod koji se prevodi obično se zove izvorni kod, a kod dobijen transformacijom mašinski kod. I naredbe tog programa se izvršavaju izvršavanjem mašinskog koda (programski jezici kao što: C, PASCAL, FURTRAN)

Sinteza - kreiranje koda, praćeno optimizacijom

Translatori su nam: kompajleri i interpreteri (kompajleri i interpreteri se jednom rečju zovu translatori)
Podaci - su memorijski objekti sa kojima program vrši operacije

Implicitna konverzija - je konverzija pri kojoj se promenjiva pri konačnom uskladištavanju u memorijskoj lokaciji , izvrši implicitna konverzija u tip podatka za koji je i deklarisana ta promenjiva.

Npr. ako je A tipa integer, B je integer, C je float. I imamo operaciju A = B + C
Operacija sabiranja ce se izvrsiti kao za float jer je C float, ali kad dobijemo rezultat, i taj rezultat, prebacujemo u promenjivu A, koja će skladištiti taj rezultat, u slučaju da je drugog tipa (u ovom slučaju, rezultat je tipa float), izvršiće se implicitna konverzija , u integer, pre uskladištavanju u memorijskoj lokaciji promenjive A

Eksplicitna konverzija - je konverzija tipa podatka promenjive, koju programer definiše.
Primer:
INT2FLOAT (konvertuj cijeli broj u float) ,
FLOAT2INT (float u cijeli broj),
CHAR2INT (karakter u cijeli broj (broj po tabeli za karakter, ASCII tabela, Unicode tabela..) ,
INT2CHAR (cijeli broj u karakter)

Ciklus - je petlja, dio algoritma koji se ponavlja više puta, dok se uslov više ne može izvršiti

Operacija pridruživanja je , =

A = B

Operacija poređenja, < > ..

A > B
B < A

ALGORITAMSKI KORACI
(ne moraju biti uključeni svi koraci, sve zavisi od svrhe programa)

- Alokacija (deklarisanje) promenjivih (zauzimanje memorijskog prostora za promenjive)
- Unos podataka (input)
- Sekvenca (X = 2,...)
- Niz naredbi
- Selekcija (IF selection)
- WHILE, FOR petlja (WHILE, FOR loop)
- Izlaz podataka (output)

SLOZENOST ALGORITMA

- Vremenska složenost - vrijeme potrebno za izvršavanje algoritma. Broji se po operacijama u algoritmu.

- Prostorna složenost - koliko memorijskog prostora zauzimaju podaci u algoritmu

- Komunikaciona složenost - koliko je potrebno komunikacije procesora sa ostalim periferijama računara prilikom izvršavanja programa

TIPOVI PODATAKA

1. Elementarni (integer (cijeli broj), float (realan broj), character (karakter)
2. Složeni (strukture, klase (class), liste, binarno drvo, graf)

KARAKTERI

- Karakter je zapisan u binarnom obliku (kao 8 bitni binarni broj), a računar zna da ga prikazuje preko tabele koju posjeduje.
Najčešće korišćena tabela je ASCII, ili UTF-8
Karakteri, tj. brojevi po kom su poređani karakteri u tabeli, idu uvek po nekom redosledu, ako velika slova počinju od broja 30 u tabeli, onda će se završit na 56 broj. I to moćemo iskoristiti za pisanje koda.

ASCII tabela se sastoji od:
- 26 malih slova engleske abecede, redno poređanih
- 26 velikih slova
- 10 cifre (od 0 - 9)
- Znakovi matematičkih operacija, simbole, specijalni simboli (za novi red, tabulacija, nova stranica, kraj niza)

DEFINICIJE TIPOVA PODATAKA

- Elementarni tip podatka (integer (cijeli broj), float (realan broj), character (karakter)
- Niz (stringovi i matrice) - kolekcija (skup) elementarnih podataka istog tipa (integer, float, character)
    - Niz - kad se govori o nizu, misli se na niz cijelih ili realnih brojeva
    - String - je niz karaktera
    - Matrica - su dvodimenzionalni nizovi, odnosno nizovi nizova A(4,4)
- Struktura - je kolekcija (skup) podataka raznih tipova podataka (u jednoj strukturi mogu biti i integer, i float, i character)
- Klasa (class) - je kolekcija (skup) podataka i elementarnih i složenih (u class se mogu nalaziti osim elementarnih podataka i funkcije, kao i svi složeni tipovi podataka koji se mogu realizovati kroz te funkcije)

- Rekurzivna funkcija - je kada funkcija poziva sama sebe

SORTIRANJE

Tipovi sortiranja niza:

- Ponovljeni minimum
- Buble sort
- Insertion sort
- Quick sort

Razlika između algoritama sortiranja:

- Ponovljeni minimum počinje od prvog elementa niza, prolazi kroz elemente niza, da kada pronađe manji, tj. najmanji element u nizu, njega će staviti na to prvo mjesto u nizu. Pa onda pređe na drugi element u nizu, i opet prolazi kroz sve elemente, da proveri ima li koji manji element u celom nizu od tog drugog.
Pri prvom pronalaženju, makar malo manjeg, odma im zameni mjesta, ali i dalje ide do kraja niza da vidi ima li ipak još manji

- Buble sort radi tako što upoređuje svaka dva susedna člana niza i zamenjuje im mesta. Prolazi se kroz niz elemenata sve dok se ne izvrši nijedna zamena tj. elementi su sortirani u odgovarajućem poretku (rastućem).

- Insertion sort uzima jedan ulazni element pri svakom prolasku i povećava sortiranu izlaznu listu. Pri prolasku, sortiranje umetanjem uklanja jedan element iz ulaznih podataka, pronalazi mesto gde pripada taj element u sortiranoj listi i stavlja ga tamo. Ponavlja prolaske sve dok ne ostane nijedan ulazni element.

Sortiranje se obično obavlja u mestu, prolazeći kroz niz, povećavanjem sortirane liste. Na svakom prolasku kroz niz, proverava vrednost ulaznog podatka, upoređujući ga sa najvećom vrednošću niza u poslednjoj proveri. Ako je veći ostavlja element na mestu i prelazi na sledeći, ako je manji nađe odgovarajuću poziciju u nizu, pomera sve veće vrednosti da napravi prostor i ubacuje element na ispravno mesto.

LISTE

- Liste su samoreferentne strukture. Odnosno, strukture koje znaju koji element je iza njih.
Program počinje od glave, i ide do kraja liste, do repa, a znaće da je kraj liste ako je referenca na toj poslednjoj nula (0).
Ta referenca je onaj pokazivač *NEXT[] u pseudokodu

- Čvor liste - je element liste
- Glava liste- je prvi element u listi
- Rep liste - je poslednji element u listi

- Prednosti listi u odnosu na nizove je to što se kod listi može lakše i brže dodati ili obrisati elementi, i osim toga, liste mogu podržati više različitih tipova podataka jer su liste složeni tip podatka, dok nizovi mogu imati samo kolekciju jednog elementarnog tipa podatka

KLASE (CLASS)

- Kod struktura, promenjivima se može pristupati direktno, dok kod klase (class) promenjivima se može pristupiti samo kroz deklarisane funkcije unutar te klase (class)

- Funkcije u klasi se takođe zovu metode (method)

- Promenjive klase se nazivaju: objekti (instance)

- Konstruktor - služi za kreiranje objekata klase (kako bi se mogli koristiti u drugim funkcijama). Stavlja joj se ime kao i sama klasa, bez argumenata

- Mutatori - služe za postavljanje vrednosti objektima (takođe se zovu setters), stavljaju se argumenti, gde će argumenti predstavljati u koje objekte (promenjive klase) će biti upisan odgovarajući argument ove funkcije (redosled stavljanja argumenata i upisivanja u odgovarajuće objekte je važan)

- Inspektori - služe za čitanje vrednosti objekata (takođe se zovu getters)

- Destruktori - su funkcije koje uništavaju kreirane objekte (dealokacija objekata). Ali, uglavnom programski jezik, ima program koji se postara za to (garbage collector) koji očisti objekte koji se više ne koriste i neće se koristiti u budućim metodama

- CONST - stavljamo u našem pseudokodu da označimo da funkcija neće menjati objekte klase, nego će ih samo moći čitati

- Prijateljske funkcije i klase - je da se omogući drugoj klasi ili funkciji da vidi privatne podatke (članove, objekte) te klase. (ovo se ustvari zove: protected iako u naš pseudokod se zove friend)

- Static - je da označimo zajednički član (objekat) klase. Znači da pripada celoj klasi, i bilo koja funkcija ga može izmeniti bez prethodnog pozivanja tog objekta

- Nasleđivanje - kada druga novokreirana klasa, nasledi već nameštene metode i članice (objekte) iz neke druge klase. Tako možemo koristiti objekte te druge klase u našoj trenutnoj klasi, objekti iz druge klase koji nam trebaju. (ovo se zove: implements)

- Preklapanje konstruktora - je kada se definiše više konstruktora (ovo takođe ga nazivaju i: konstruktor kopije)

Npr: a:complex
b:complex(2)

Ovo preklapanje konstruktora služi za kreiranje samo tog jednog objekta, umesto svih kao što je slučaj za taj default konstruktor što radi. Tj. ovakav, preklopljeni konstruktor nam služi za implicitnu konverziju podataka.

Npr. imamo taj jedan preklopljeni konstruktor osim onog običnog koji kreira sve objekte, sa ovim preklopljenim konstruktorom, možeš izvršiti implicitnu konverziju podatka, tj. da je korisnik slobodan da unese drugi tip podatka, iako će to u samom tom (preklopljenom) konstruktoru da bude prevedeno na tip podatka koji taj član klase (objekat) podržava

Član klase je integer C
I kreiramo preklopljeni konstruktor, koji će primati float

KLASA(A:float)

i realizacija tog preklopljenog konstruktora je:

KLASA (A : float)
C = A

i to je to, iako je korisnik bio slobodan da ubaci u float, to ce ovim konstruktorom, biti prebaceno u integer jer je C integer.
I zato i služi taj preklopljeni konstruktor, i takođe kao što pomenuh već, da se kreira objekat samo za tu jednu promenjivu (članicu klase)

- I kod funkcija, ako stavljaš, prosleđuješ niz, pa makar i te same klase, i dalje isto važi pravilo, da treba da staviš i dužinu niza. čak iako je ta funkcija definisana u samoj toj klasi. to treba.
I return ide na te funkcije !! jedino ne ide na konstruktore, mutatore (setters) i inspektore (getters) return, ali na ostale funkcije klase, i dalje ide return, ili vraca neki elementarni tip podatka, ili ne vraca nista, sve obavi u samom toj funkciji.

PAR PITANJA, I ODGOVORI

- Koje su prednosti korišćenja potprograma (funkcija) u programiranju?

Prednosti korišćenja potprograma (funkcija) u programiranju, je da u funkciji napišemo neki program koji treba nešto da odradi, i da gde god nam treba, samo pozovemo taj potprogram, i proslijedimo mu argumente koje su potrebne da taj potprogram izvrši

- Objasniti razlike u prosleđivanju argumenata funkciji po vrijednosti i po referenci?

Kada prosleđujemo argument funkciji po vrijednosti, šta god mi radimo u toj funkciji sa tom promenjivom, to neće uticati na promenjivu koja je u glavnom programu
Dok kada prosleđujemo argument funkciji po referenci, šta god mi radimo, u toj funkciji sa sa tom promenjivom, to će direktno uticati na promenjivu koja je u glavnom programu, imeniće je u glavnom programu ako se ta promenjiva u funkciji izmeni

- Objasniti na koji način funkcija može da vrati više od jednog rezultata tj. da promijeni više promjenjivih koje pripadaju glavnom dijelu algoritma?

Tako što ćemo samo jednu promenjivu (koja je elementarnog tipa) vratiti kroz RETURN, a za ostale promenjive ćemo koristiti poziv po referenci (&A:integer...)