Strukture podataka

*- zadatak 1 -*

1. Osnovni pojmovi i dijelovi stuktura podataka

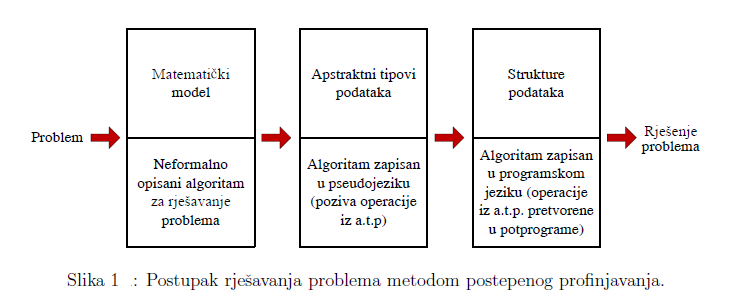
Strukture podataka su skup varijabla u nekom programu zajedno s vezama između tih varijabli s namjerom da omogući pohranjivanje odredenih podataka te efikasno izvršavanje odredenih operacija s tim podacima (upisivanje, promjena, čitanje, traženje po nekom kriteriju, itd).U programskom jeziku C++ strukture alociramo statički ili dinamički, a razlika među this struktura da dinamičkom alokacijom se njihova lokacija u memoriji ne zadaje tokom prevođenja programa već tijekom izvršavanja.

Strukture podataka ipak nisu same sebi svrši shodne jer bez operacija koje vršimo nad njim nisu nam od koristi, a te operacije definiramo pomoću algoritama. Algoritam je konačan niz postupaka od kojih svakih od njih ima svoje značenje i može se izvršiti u konačnom vremenu.

Tip podataka je skup vrijednosti koje neki podatak može poprimiti. Prema tipu možemo nad podatkom vršiti određene operacije i njime manipulirati. Većina objekata koje koristimo u stvarnom životu se definiraju kroz različite tipove podataka te za njih jedino agregatno stanje u programiranju su nam abstraktni tipovi podataka.   
 Apstraktni tip podataka (ATP) zadaje se navodenjem jednog ili više tipova podataka, te jedne ili više operacija (funkcija). U implementaciji programski operandi i rezultati navedenih operacija su podaci navedenih tipova. Među tipovima postoji jedan istaknuti po

kojem cijeli apstraktni tip podataka dobiva ime.

Prije impelemntacije moramo neformalno opisati algoritam za riješenje problema, algoritam zapisati u pseudojeziku te ga prevesti u programski jezik (u našem slučaju C++) , što je vidljivo na Slika 1.



Na slici Slika 1 jako dobro je vidljiva uloga apstraktnih tipova podataka, struktura podataka te algoritama u postupku rješavanja problema.

Prije nego krenemo dalje ukratko ćemo definirati gradivne elemente struktura podataka te primjer implementacije/sintakse unutar C++ programskog jezika.

a) varijabla

* + imenovani alocirani prostor u memoriji
  + ima tip, ime i vrijednost (promjenjiva)
  + C++, C# sintakse:

<tip podataka><ime varijable>; npr.: int a;

b) polje

* više memorijski uzastopno povezanih varijabla istog tipa
* varijable se u polju nazivaju elementima polja
* svakom elementu polja se pristupa preko njegovog indeksa
* veličina polje je unaprijed zadana i nepromjenjiva
* C++ implementacija

: <tip podataka><ime polja>[<broj elemenata>]; npr.: char h[10];

c) slog (stack)

* + predstavlja mehanizam udruživanja varijabla sličnom polju
  + podaci se također nalaze na uzastopnim lokacijama u memoriji, ali ne moraju biti istog tipa
  + funkcionira prema principu LIFO(Last in first out) tj. element koji se zadnji upisuje u stog se prvi čita sa stoga
  + tip, redoslijed i duljina stoga su unaprijed zadani i nepromijenjivi
  + jedan zapis u slogu se naziva komponenta sloga
  + C++ implementacija:

class stack

{

int s[7];

// Operacije koje se vrše nad stogom

public:

void push(int);

void pop();

void top();

void empty();

void show();

};

d) pointer

* + prestavlja pokazivačku varijablu koja pokazuje na memorijsku lokaciju druge varijable
  + u programu služi za uspostavljanje veze između dijelova strukture te za dinamičku alokaciju
  + C++ sintaksa:

<tip podataka> \*<naziv pokazivača>

f) kursor

* + također služi za povezivanje dijelova strukture, ali je tipa integer
  + njegova vrijednost je zapravo indeks polja
  + C++ sintaksa ista kao i kod varijable

2. ATP polinom

ATP polinom je struktura podataka koja se sastoji od jednodimenzionalnog polja i jednog kursora cjelobrojne vrijednosti koji označava duljinu polinoma.

Zapis (slog) ATP polinoma u C++:

struct pol

{

double values[1000];

int en;

};

typedef struct pol \*polinom;

Algoritmi koji se implementiraju zajedno s ovim polinomima te vrše određene operacije nad istim:

Zero(p)

* + funkcija pomoću koje iniciramo nul-polinom

IsZero(p)

* + funkcija koja provjerava da li je trenutni polinom nul-polinom

Coef(p, pot)

* + funkcija vraća koeficijent u polinomu uz zadanu potenciju

Attach(p, pot, val)

* + funkcija koja pridružuje vrijednost od određenom koeficijentu

Degree(p)

* + funkcija koja vraća stupanj polinoma

Add(p1, p2, p3)

* + funkcija za zbrajanje dva polinoma

Mult(p1, p2, p3)

* + funkcija za množenje dva polinoma

Sada ćemo prikazati vremensku složenost istih funkcija:

Zero(p) - O(1) - **konstantan algoritam** [O(1)] - vrijeme izvođenja je ograničeno konstantom i  ne ovisi o ulaznim parametrima

IsZero(p) - O(1)

Degree(p) - O(1)

Coef(p, pot) - O(1)

Attach(p, pot, val) - O(n) -  **linearni algoritam** [O(n)] - povećanjem količine radnji, povećava se i vrijeme obrade.

Add(p1, p2, p3) - O(n)

Mult(p1, p2, p3) - O(n2) - **stupanjski algoritam** [O(nm)] -  najčešće nastaju kada se koriste ugniježđene

*for* petlje.

**Literatura:**

1. Dr. sc. D. Radošević, “Programiranje 2”, TIVA 2007.

2. D. E. Knuth, “Fundamental Algorithm”, Addison-Wesley 1997.

3. ELF, dostupno na: <http://elf.foi.hr/course/view.php?id=202>, materijali iz “Struktura podataka”

4. R. Manger, “Strukture podataka i algoritmi”, Zagreb 2013.,  dostupno 6.11.2013 na: <http://web.studenti.math.pmf.unizg.hr/~manger/spa/skripta.pdf>

5. Ž. Vrba, M. Bulaja, “Strukture podataka i algoritimi” - (v.2005-02-21) - dostupno 6.11.2013 na:

<http://zvrba.net/writings/spa.pdf>