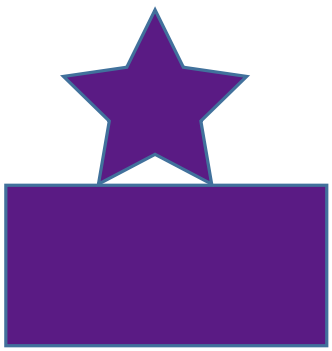
Adrian Romeo Șișiroi

Limbajul de programare C++

-teorie și aplicații-



CUPRINS

Sfaturi utile:

1) Să se folosească cu preponderența calculatoarele desktop în detrimental calculatoarelor laptop deoarece protejează mai bine ochii
2) Calculatorul să fie așezat la birou astfel încât lumina să vină din lateral
3) Timpul petrecut în fața ecranului să fie de aproximativ o jumătate de oră urmat de pauze de minim 10 minute
4) Programele de calculator se schițează, se incearcă să se rezolve pe suport de hârtie si mai apoi se implementează, se finisează, se corectează erorile în mediul de programare, la calculator

Spor la lucru!

Autorul

Algoritmi

Notiunea de algoritm

Ciclul de viata al unui program cuprinde fazele de:

- Analiza
- Proiectare
- Codificare
- Testare
- Intretinere.

In faza de proiectare se stabileste **algoritmul**, adica pasii succesivi de rezolvare a problemei. Denumirea de algoritm vine de la numele matematicianului si astrologului arab *Al-Khowarizmi* (secolul IX).

Un **algoritm** în matematică şi informatică reprezintă o metodă de rezolvare a unei probleme alcătuită din paşi reprezentaţi de operaţii/acţiuni.

Proprietăţile algoritmului:

- Finitudinea – furnizarea rezultatelor după un număr finit de paşi
- Claritatea – precis, fără ambiguităţi
- Generalitatea- acoperă o clasă generală de probleme, nu una particulară
- Corectitudinea – furnizează soluţii corecte
- Unicitatea-furnizează soluţii unice pentru aceleaşi date de intrare, indiferent de numărul de efectuări ale algoritmului
- Eficienţa-numărul de paşi parcurşi este cât se poate de mic
- Optimalitatea – algoritmul urmează calea directă de rezolvare
- Verificabilitatea –fiecare pas poate fi verificat
- Completitudinea- sunt tratate şi cazurile particulare ale unei probleme generale

Algoritmul secvenţial este un algoritm în care acţiunile se execută liniar, în ordinea în care apar, una după alta , de la început până la sfârşit.

Algoritmul decizional este un algoritm în care acţiunile care se execută se stabilesc în baza unei decizii în urma evaluării stabilindu-se pe ce ramificaţie se marge mai departe.

Algoritmul repetitiv este un algoritm în care acţiunile care se execută se repetă .

Exemple de algoritmi din viaţa reală:

- **Algoritmi secvenţiali**

- Algoritmul preparării unei căni de ceai
- Algoritmul programului de dimineaţa

- **Algoritmi bazaţi pe decizii**

- Algoritmul de traversare a unei străzi cu semafor pe zebră (decizia este luată în funcţie de culoarea semaforului verde sau roşie)
- Algoritmul parcurgerii unui drum care la un moment dat se bifurcă (decizia de a urma o cale spre stânga sau spre dreapta este luată în funcţie de anumite elemente)

- **Algoritmi bazaţi pe repetiţii**

- Algoritmul de dirijare a unei orchestre (deseori unele mişcări ale mâinii sunt repetitive)

Algoritmi de complexitate mai mare(pot conţine acţiuni secvenţiale, decizionale şi repetitive):

- Algoritmul de construcţie a unui automobil, case, bloc de locuinţe
- algoritmul de folosire a unei maşini unelte(citind manualul de folosire)

Descriere algoritmi din viaţa reală:

Exemple:

Algoritm secvenţial

- *Algoritmul preparării unei căni de ceai*

Ce vom folosi? Plic cu plante pentru ceai, zahăr, apă, fierbător, cană

Rezultat:ceai de plante

Cum procedăm?

Mergem la bucătărie, luăm fierbătorul, punem apă în el şi îl pornim. Luăm o cană, punem în ea pliculeţul de ceai şi zahărul. După ce a început să fiarbă apa, o turnăm în cană, amestecăm şi aşteptăm 4 minute. Ceaiul nostru este gata.

Acesta este algoritmul descris în limbaj natural. Haideţi acum să vedem cum ar arăta algoritmul nostru într-o manieră mai structurată, pe paşi dar tot în limbaj natural

Pasul 0:Început

Pasul 1: Mergem la bucătărie

Pasul 2: Luăm fierbătorul

Pasul 3: Punem apă în fierbător

Pasul 4: Pornim fierbătorul

Pasul 5:Punem într-o cană zahărul şi pliculeţul de ceai

Pasul 6: Aşteptăm până fierbe apa

Pasul 7: Turnăm apa fierbinte în cana pregătită anterior

Pasul 8: Amestecăm şi aşteptăm patru minute

Pasul 9:Ceaiul este gata

Pasul 10 Sfârşit

Algoritmul decizional

- *Algoritmul de traversare a unei străzi cu semafor pe zebră*

Ce vom folosi? Picioarele

Rezultat: Traversarea sau nu a străzii

Cum procedăm?

Mergem pe stradă . În momentul în care ajung la trecerea de pietoni iau decizia de a traversa sau nu strada. Dacă culoarea semaforului este verde atunci, DA traversăm strada altfel, Nu traversăm strada. În funcţie de culoarea semaforului am traversat sau nu strada.

Acesta este algoritmul descris în limbaj natural. Haideţi acum să vedem cum ar arăta algoritmul nostru într-o manieră mai structurată, pe paşi dar tot în limbaj natural

Pasul 0:Început

Pasul 1: Mergem pe stradă.

Pasul 2: În momentul în care ajung la trecerea de pietoni.

Dacă culoarea semaforului este verde atunci, DA traversăm strada

Altfel, Nu traversăm strada

Pasul 3: Sfârşit

Algoritmul repetitiv

- *Algoritmul de dirijare a unei orchestre*

Ce vom folosi? Mâinile

Rezultat: Interpretat o melodie după o partitură

Cum procedăm?

Citim după portativ .La notele înalte se ridică mâinile/mâna .La notele joase se coboară mâinile/mâna. Se repetă acţiunile precedente până am terminat notele muzicale ale piesei.

Acesta este algoritmul descris în limbaj natural. Haideţi acum să vedem cum ar arăta algoritmul nostru într-o manieră mai structurată, pe paşi dar tot în limbaj natural

Pasul 0: Început
Pasul 1: Citim după portativ
Pasul 2: La notele înalte se ridică mâinile/mâna
Pasul 3:La notele joase se coboară mâinile/mâna
Pasul 4 Se repetă paşii 1 şi 2
Pasul 5: Sfârşit

Algoritmii poate fi specificati:

- In limbaj natural
- In pseudocod
- Ca scheme logice.

Date

Algoritmii lucrează cu date, adică cu valori , care sunt reţinute de variabile.

Datele sunt obiectele cu care lucrează orice algoritm. Datele sunt caracterizate prin **nume**, **tip** şi **valoare**.

Clasificarea datelor

Datele se clasifică:

I)În funcţie de momentul utilizării lor în algoritm

-Date de intrare

-Date intermediare

-Date de ieşire

II)În funcţie de tip:

Clasificarea tipurilor de date:

- **Tipuri fundamentale(standard sau predefinite)**
- **Tipuri derivate**

A. Tipuri fundamentale (elementare)

Clasificarea tipurilor fundamentale:

- Tipul void
- Tipuri aritmetice(char ,int, float, double)
- Logice(Adevărat(True),False(Fals))
- Şiruri de caractere

Gama tipurilor aritmetice poate fi extinsă prin setul de modificatori de tip şi anume: signed(cu semn), unsigned (fără semn), short(scurt), long(lung)