Curs Algoritmi Fundamentali (AF)
Noţiuni introductive

Lect. dr. Alexandra Băicoianu

Universitatea *Transilvania* din Braşov
Facultatea de Matematică şi Informatică

2020

Agenda

Identificare actorilor şi colectarea cerinţelor (pentru AF)

O cerinţă (eng. requirement) reprezintă un element de funcţionalitate pe care sistemul trebuie sa îl ofere sau o constrângere pe care trebuie să o îndeplinească.
Actorii (eng. actor) sunt roluri jucate de diverse persoane sau sisteme informatice şi care interacţionează cu sistemul informatic aflat în dezvoltare.

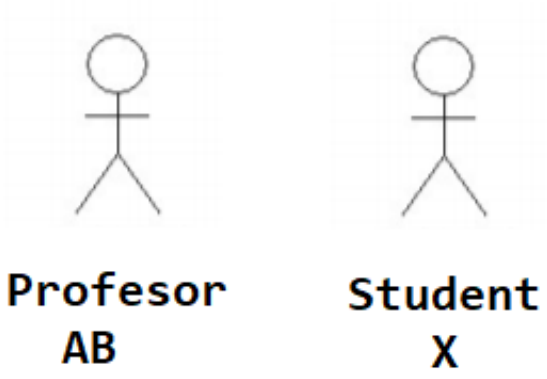

Figure: Diverşi actori în procesul de ÎNVĂŢARE (AF)

Colectarea cerinţelor

- Limbaj de programare utilizat (Dacă C++ atunci VS minim 2017)

Colectarea cerinţelor

- Limbaj de programare utilizat (Dacă C++ atunci VS minim 2017)
- Ponderi AF: 1 - laborator (25%), 2 - proiect per echipa de 4 persoane (15%), 3 - examen scris (60%)

Colectarea cerinţelor

- Limbaj de programare utilizat (Dacă C++ atunci VS minim 2017)
- Ponderi AF: 1 - laborator (25%), 2 - proiect per echipa de 4 persoane (15%), 3 - examen scris (60%)
- Laboratoare: Alexandra Băicoianu (IA3), Luciana Majercsik (I1, I2, I3, IA2), Radu Todor (IA1, I4)

Colectarea cerinţelor

- Limbaj de programare utilizat (Dacă C++ atunci VS minim 2017)
- Ponderi AF: 1 - laborator (25%), 2 - proiect per echipa de 4 persoane (15%), 3 - examen scris (60%)
- Laboratoare: Alexandra Băicoianu (IA3), Luciana Majercsik (I1, I2, I3, IA2), Radu Todor (IA1, I4)
- Clean Code TIPS (Labs) + Clean Code TIPS - vezi pagină curs elearning

Ce este un algoritm?

Un algoritm este compus dintr-o succesiune finită de operaţii sau procese. Aplicarea unui algoritm presupune executarea acestora într-o anumită ordine. Algoritmii pe care îi descriem trebuie să respecte câteva restricţii:

- să fie cât mai generali, adică să rezolve o clasă specifică de probleme de acelaşi tip;

Ce este un algoritm?

Un algoritm este compus dintr-o succesiune finită de operaţii sau procese. Aplicarea unui algoritm presupune executarea acestora într-o anumită ordine. Algoritmii pe care îi descriem trebuie să respecte câteva restricţii:

- să fie cât mai generali, adică să rezolve o clasă specifică de probleme de acelaşi tip;
- să fie corect, de cele mai multe ori corectitudinea fiind demonstrată pe cazul general şi pe cazuri particulare;

Ce este un algoritm?

Un algoritm este compus dintr-o succesiune finită de operaţii sau procese. Aplicarea unui algoritm presupune executarea acestora într-o anumită ordine. Algoritmii pe care îi descriem trebuie să respecte câteva restricţii:

- să fie cât mai generali, adică să rezolve o clasă specifică de probleme de acelaşi tip;
- să fie corect, de cele mai multe ori corectitudinea fiind demonstrată pe cazul general şi pe cazuri particulare;
- să se încheie într-un anumit timp finit (număr finit de paşi indiferent de datele de intrare);

Ce este un algoritm?

Un algoritm este compus dintr-o succesiune finită de operaţii sau procese. Aplicarea unui algoritm presupune executarea acestora într-o anumită ordine. Algoritmii pe care îi descriem trebuie să respecte câteva restricţii:

- să fie cât mai generali, adică să rezolve o clasă specifică de probleme de acelaşi tip;
- să fie corect, de cele mai multe ori corectitudinea fiind demonstrată pe cazul general şi pe cazuri particulare;
- să se încheie într-un anumit timp finit (număr finit de paşi indiferent de datele de intrare);
- să asigure unicitatea rezultatelor, adică pentru acelaşi set de date de intrare să obţinem acelaşi rezultat indiferent de context;

Ce este un algoritm?

Un algoritm este compus dintr-o succesiune finită de operaţii sau procese. Aplicarea unui algoritm presupune executarea acestora într-o anumită ordine. Algoritmii pe care îi descriem trebuie să respecte câteva restricţii:

- să fie cât mai generali, adică să rezolve o clasă specifică de probleme de acelaşi tip;
- să fie corect, de cele mai multe ori corectitudinea fiind demonstrată pe cazul general şi pe cazuri particulare;
- să se încheie într-un anumit timp finit (număr finit de paşi indiferent de datele de intrare);
- să asigure unicitatea rezultatelor, adică pentru acelaşi set de date de intrare să obţinem acelaşi rezultat indiferent de context;
- în plus, un algoritm ar trebui să fie flexibil, eficient, lizibil şi modular;

Ce proprietăţi are un algoritm?

- Corectitudine = proprietatea de a respecta specificaţiile şi a da rezultate corecte;

Ce proprietăţi are un algoritm?

- Corectitudine = proprietatea de a respecta specificaţiile şi a da rezultate corecte;
- Extensibilitate = posibilitatea adaptării la unele schimbări în specificaţie;

Ce proprietăţi are un algoritm?

- Corectitudine = proprietatea de a respecta specificaţiile şi a da rezultate corecte;
- Extensibilitate = posibilitatea adaptării la unele schimbări în specificaţie;
- Robusteţe = abilitatea de a recunoaşte situaţiile în care problema ce se rezolvă nu are sens şi de a se comporta în consecinţă;

Ce proprietăţi are un algoritm?

- Corectitudine = proprietatea de a respecta specificaţiile şi a da rezultate corecte;
- Extensibilitate = posibilitatea adaptării la unele schimbări în specificaţie;
- Robusteţe = abilitatea de a recunoaşte situaţiile în care problema ce se rezolvă nu are sens şi de a se comporta în consecinţă;
- Reutilizabilitate = posibilitatea reutilizării întregului program sau a unor părţi din el în alte aplicaţii;

Ce proprietăţi are un algoritm?

- Corectitudine = proprietatea de a respecta specificaţiile şi a da rezultate corecte;
- Extensibilitate = posibilitatea adaptării la unele schimbări în specificaţie;
- Robusteţe = abilitatea de a recunoaşte situaţiile în care problema ce se rezolvă nu are sens şi de a se comporta în consecinţă;
- Reutilizabilitate = posibilitatea reutilizării întregului program sau a unor părţi din el în alte aplicaţii;
- Compatibilitate = uşurinţa de combinare cu alte produse;

Ce proprietăţi are un algoritm?

- Corectitudine = proprietatea de a respecta specificaţiile şi a da rezultate corecte;
- Extensibilitate = posibilitatea adaptării la unele schimbări în specificaţie;
- Robusteţe = abilitatea de a recunoaşte situaţiile în care problema ce se rezolvă nu are sens şi de a se comporta în consecinţă;
- Reutilizabilitate = posibilitatea reutilizării întregului program sau a unor părţi din el în alte aplicaţii;
- Compatibilitate = uşurinţa de combinare cu alte produse;
- Portabilitate = posibilitatea de folosire a produsului pe alte sisteme de calcul, diferite de cel pe care a fost conceput;

Ce proprietăţi are un algoritm?

- Corectitudine = proprietatea de a respecta specificaţiile şi a da rezultate corecte;
- Extensibilitate = posibilitatea adaptării la unele schimbări în specificaţie;
- Robusteţe = abilitatea de a recunoaşte situaţiile în care problema ce se rezolvă nu are sens şi de a se comporta în consecinţă;
- Reutilizabilitate = posibilitatea reutilizării întregului program sau a unor părţi din el în alte aplicaţii;
- Compatibilitate = uşurinţa de combinare cu alte produse;
- Portabilitate = posibilitatea de folosire a produsului pe alte sisteme de calcul, diferite de cel pe care a fost conceput;
- Eficienţă = măsura în care sunt bine folosite resursele sistemului de calcul;

Ce proprietăţi are un algoritm?

- Corectitudine = proprietatea de a respecta specificaţiile şi a da rezultate corecte;
- Extensibilitate = posibilitatea adaptării la unele schimbări în specificaţie;
- Robusteţe = abilitatea de a recunoaşte situaţiile în care problema ce se rezolvă nu are sens şi de a se comporta în consecinţă;
- Reutilizabilitate = posibilitatea reutilizării întregului program sau a unor părţi din el în alte aplicaţii;
- Compatibilitate = uşurinţa de combinare cu alte produse;
- Portabilitate = posibilitatea de folosire a produsului pe alte sisteme de calcul, diferite de cel pe care a fost conceput;
- Eficienţă = măsura în care sunt bine folosite resursele sistemului de calcul;
- Claritate = uşurinţa citirii şi înţelegerii coţinutului programului, a structurilor din care este compus şi a rolului denumirilor şi părţilor sale.

Studiul unei probleme date

Studiul unei probleme cuprinde mai multe etape:

1. Formularea clară a problemei şi a cerinţelor;

Aşadar, studiul algoritmilor cuprinde mai multe aspecte: stabilirea contextului practic pentru algoritmul în cauză, elaborarea algoritmului, implementarea lui într-un limbaj de programare, validarea, analiza şi testarea programelor.

Studiul unei probleme date

Studiul unei probleme cuprinde mai multe etape:

1. Formularea clară a problemei şi a cerinţelor;
2. Modelarea formală a problemei;

Aşadar, studiul algoritmilor cuprinde mai multe aspecte: stabilirea contextului practic pentru algoritmul în cauză, elaborarea algoritmului, implementarea lui într-un limbaj de programare, validarea, analiza şi testarea programelor.

Studiul unei probleme date

Studiul unei probleme cuprinde mai multe etape:

1. Formularea clară a problemei şi a cerinţelor;
2. Modelarea formală a problemei;
3. Clarificarea sau alegerea algoritmilor necesari;

Aşadar, studiul algoritmilor cuprinde mai multe aspecte: stabilirea contextului practic pentru algoritmul în cauză, elaborarea algoritmului, implementarea lui într-un limbaj de programare, validarea, analiza şi testarea programelor.

Studiul unei probleme date

Studiul unei probleme cuprinde mai multe etape:

1. Formularea clară a problemei şi a cerinţelor;
2. Modelarea formală a problemei;
3. Clarificarea sau alegerea algoritmilor necesari;
4. Implementarea algoritmilor într-un limbaj de programare;

Aşadar, studiul algoritmilor cuprinde mai multe aspecte: stabilirea contextului practic pentru algoritmul în cauză, elaborarea algoritmului, implementarea lui într-un limbaj de programare, validarea, analiza şi testarea programelor.

Studiul unei probleme date

Studiul unei probleme cuprinde mai multe etape:

1. Formularea clară a problemei şi a cerinţelor;
2. Modelarea formală a problemei;
3. Clarificarea sau alegerea algoritmilor necesari;
4. Implementarea algoritmilor într-un limbaj de programare;
5. Testarea funcţionalităţii;

Aşadar, studiul algoritmilor cuprinde mai multe aspecte: stabilirea contextului practic pentru algoritmul în cauză, elaborarea algoritmului, implementarea lui într-un limbaj de programare, validarea, analiza şi testarea programelor.

Studiul unei probleme date

Studiul unei probleme cuprinde mai multe etape:

1. Formularea clară a problemei şi a cerinţelor;
2. Modelarea formală a problemei;
3. Clarificarea sau alegerea algoritmilor necesari;
4. Implementarea algoritmilor într-un limbaj de programare;
5. Testarea funcţionalităţii;
6. Mentenanţa rezolvării.

Aşadar, studiul algoritmilor cuprinde mai multe aspecte: stabilirea contextului practic pentru algoritmul în cauză, elaborarea algoritmului, implementarea lui într-un limbaj de programare, validarea, analiza şi testarea programelor.

Conteză cum scriem cod?

- Keep it simple, stupid (KISS)

Conteză cum scriem cod?

- Keep it simple, stupid (KISS)
- Don’t repeat yourself (DRY)

Conteză cum scriem cod?

- Keep it simple, stupid (KISS)
- Don’t repeat yourself (DRY)
- Be precise

Conteză cum scriem cod?

- Keep it simple, stupid (KISS)
- Don’t repeat yourself (DRY)
- Be precise
- Only one responsibility

Conteză cum scriem cod?

- Keep it simple, stupid (KISS)
- Don’t repeat yourself (DRY)
- Be precise
- Only one responsibility
- Describe the intention

Conteză cum scriem cod?

- Keep it simple, stupid (KISS)
- Don’t repeat yourself (DRY)
- Be precise
- Only one responsibility
- Describe the intention
- Use explanatory names

Conteză cum scriem cod?

- Keep it simple, stupid (KISS)
- Don’t repeat yourself (DRY)
- Be precise
- Only one responsibility
- Describe the intention
- Use explanatory names
- Avoid abbreviation

Conteză cum scriem cod?

- Keep it simple, stupid (KISS)
- Don’t repeat yourself (DRY)
- Be precise
- Only one responsibility
- Describe the intention
- Use explanatory names
- Avoid abbreviation
- Use names from the domain language

Conteză cum scriem cod?

- Keep it simple, stupid (KISS)
- Don’t repeat yourself (DRY)
- Be precise
- Only one responsibility
- Describe the intention
- Use explanatory names
- Avoid abbreviation
- Use names from the domain language
- Delete commented & unused code

Conteză cum scriem cod?

- Keep it simple, stupid (KISS)
- Don’t repeat yourself (DRY)
- Be precise
- Only one responsibility
- Describe the intention
- Use explanatory names
- Avoid abbreviation
- Use names from the domain language
- Delete commented & unused code
- Keep functions/classes short

Conteză cum scriem cod?

- Keep it simple, stupid (KISS)
- Don’t repeat yourself (DRY)
- Be precise
- Only one responsibility
- Describe the intention
- Use explanatory names
- Avoid abbreviation
- Use names from the domain language
- Delete commented & unused code
- Keep functions/classes short
- Avoid magic numbers

Conteză cum scriem cod?

- Keep it simple, stupid (KISS)
- Don’t repeat yourself (DRY)
- Be precise
- Only one responsibility
- Describe the intention
- Use explanatory names
- Avoid abbreviation
- Use names from the domain language
- Delete commented & unused code
- Keep functions/classes short
- Avoid magic numbers
- Simplify complex conditional expressions

Câteva probleme

1. Să se scrie o funcţie/subprogram/metodă care returnează suma, dar şi produsul a două numere întregi date;

Câteva probleme

1. Să se scrie o funcţie/subprogram/metodă care returnează suma, dar şi produsul a două numere întregi date;
2. Se consideră un tablou cu n elemente având valori din 1, ..., *n*. Tabloul poate avea toate elementele distincte sau poate exista o pereche de elemente cu aceeaşi valoare (o singură astfel de pereche). Să se verifice dacă elementele tabloului sunt toate distincte sau dacă exist]ă o pereche de elemente distincte. Cât este complexitatea algoritmului propus?