Complemente de C++

Autori: Vlad Huțanu, Tudor Sorin

CUPRINS

Lecția 13. Constructori de copiere
Lecția 14. Supraîncarcarea operatorului "[]"
Lecția 15. Supraîncarcarea operatorului "()"
Lecția 16. Membri statici ai unei clase
Lecția 17. Tratarea excepțiilor prin utilizarea claselor
Lecția 18. Clase derivate
Lecția 19. Accesul la membrii unei clase (modificatorii "public","protected","private")
Lecția 20. Comportamentul constructorilor şi desctructorilor în cazul derivării claselor
Lecția 21. Pointeri către obiecte
Lecția 22. Metode virtuale
Lecția 23. Metode virtuale pure
Lecția 24. Clasa "string" (din STL, pentru a lucra uşor cu şirurile de caractere)

Partea a IV-a. Inițiere în programarea generică

Lecția 1. Ce este programarea generică ?
Lecția 2. Funcții template
Lecția 3. Clase template
Lecția 4. Derivarea în cazul claselor template

Partea a V-a. Inițiere în STL (Standard Template Library)

Lecția 1. Ce este STL ?
Lecția 2. Mai mult despre containere
Lecția 3. Mai mult despre iteratori
Lecția 4. Un exemplu mult simplificat pentru înțelegerea noțiunilor de container, iterator şi algoritm
Lecția 5. Clasa **vector<T>**
Lecția 6. Clasa **list<T>**
Lecția 7. Clasa **set<T, Comp>**
Lecția 8. Clasa **multiset<T,Comp>**
Lecția 9. Clasa **map<Cheie, Data, Compar>**
Lecția 10. Clasa **stack<T>** (stiva)
Lecția 11. Clasa **queue<T>** (coada)
Lecția 12. Clasa **priority_queue<T,Sequence,Compar>** (coada)
Lecția 13. Cum se scrie un algoritm ?
Lecția 14. Algoritmii şi predicatele
Lecția 15. Algoritmii şi elementele de asociere
Lecția 16. Algoritmul **for_each()**
Lecția 17. Algoritmi de tip **find**
Lecția 18. Algoritmi de tip **count**
Lecția 19. Algoritmi de tip **search**
Lecția 20. Algoritmi de tip **mismatch** si **equal**
Lecția 21. Algoritmi de tip **copy**
Lecția 22. Algoritmi de tip **replace**
Lecția 23. Algoritmi de tip **remove**
Lecția 24. Algoritmi de tip **transform**