

Complemente de C++

Autori: Vlad Huțanu, Tudor Sorin

CUPRINS

Partea I. Noțiuni introductive

- Lecția 1.** Istoric C/C++
- Lecția 2.** GNU Project și Free Software Foundation
- Lecția 3.** Scrieți programe utilizând **MinGW Developer Studio**
- Lecția 4.** Scrieți programe utilizând **DJGPP**

Partea a II-a. Elemente de limbaj

- Lecția 1.** Pointeri
- Lecția 2.** Referințe
- Lecția 3.** Modificatorul **"const"**
- Lecția 4.** Legatura între masive și pointeri
- Lecția 5.** Operații cu pointeri
- Lecția 6.** Expresii indexate
- Lecția 7.** Alocarea dinamică a memoriei
- Lecția 8.** Parametri pentru program
- Lecția 9.** Supraîncarcarea funcțiilor
- Lecția 10.** Funcții cu un număr variabil de parametri
- Lecția 11.** Pointeri către funcții
- Lecția 12.** Spații de nume
- Lecția 13.** Instrucțiunile **"break"** și **"continue"**
- Lecția 14.** Excepții

Partea a III-a. Inițiere în programarea orientată pe obiecte

- Lecția 1.** Ce este programarea orientată pe obiecte ?
- Lecția 2.** Clase
- Lecția 3.** Constructori
- Lecția 4.** Destructor
- Lecția 5.** Copierea obiectelor
- Lecția 6.** Funcții prietene
- Lecția 7.** Supraîncarcarea operatorilor cu metode ale clasei. Generalități
- Lecția 8.** Supraîncarcarea operatorilor prin funcții prietene
- Lecția 9.** Cuvântul cheie **"this"**
- Lecția 10.** Despre supraîncarcarea unor operatori unari
- Lecția 11.** Supraîncarcarea operatorilor **"++"**, **"--"**
- Lecția 12.** Supraîncarcarea operatorului **"="**

Lecția 13. Constructori de copiere
Lecția 14. Supraîncărcarea operatorului “[]”
Lecția 15. Supraîncărcarea 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 și destructorilor 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 ușor cu șirurile 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 și 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 și predicatle
Lecția 15. Algoritmii și elementele de asociere
Lecția 16. Algoritmii **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** și **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**

Lecția 25. Algoritmi de tip **sort**
Lecția 26. Căutarea binară
Lecția 27. Ordinea lexicografică
Lecția 28. Partiționare
Lecția 29. Algoritmi de tip **fill**
Lecția 30. Algoritmi de tip **unique**