MD-2045, CHIŞINĂU, STR. STUDENȚILOR, 9/7, TEL: 022 50-99-01 | FAX: 022 50-99-05, www.utm.md

PROGRAMMATION ORIENTÉE OBJET

1. Informations sur l'unité de cours / module

Faculté	Ordinateurs, Informatique et Microélectronique				
Chaire/département	Filière Francophone Informatique, dép. Génie Logiciel et Automatique				
Cycle d'études	Études supérieures, Licence - cycle I				
Programme d'études	06 I 3. I Technologie de l'information				
Année d'étude	**Semestre**	**Type d'évaluation**	**Catégorie formative**	**Catégorie d'option**	**Crédites ECTS**
II (enseignement à plein temps);	3	E	S – Unité de cours de spécialité	O - Unité de cours obligatoire	5

2. Timpul total estimat

Nombre total d'heures dans le programme	Dont				
	Heures dans la salle de cours		Travail individuel		
	Cours	Travaux pratique/dirigés	Projet d'année	Étude du matériel théorique	Préparation de l'application
150	30	30/15		60	45

3. Prérequis pour l'accès à l'unité de cours/module

Selon le programme d'études	Langage de programation C.
Selon les compétences	Connaissances et compétences pour concevoir et développer des algorithmes et des programmes en langage C pour résoudre des problèmes à l'ordinateur.

4. Conditions de déploiement le processus éducatif pour

Cours	Pour présenter le matériel théorique en classe, il faut un tableau noir, un projecteur et un ordinateur.
Travaux pratique/dirigés	Les étudiants rédigeront des rapports selon les conditions formulées dans les indications méthodiques. La durée du soutien d'un travail pratique est une semaine après l'achèvement. La soumission tardive du document est pénalisée : -1 point pour une semaine de retard.

5. Compétences spécifiques accumulées

Compétences professionnelles	✓ Capacité de développer les étapes de résolution de problèmes à l'ordinateur. ✓ Connaissance des structures algorithmiques utilisées dans les algorithmes. ✓ Connaissance du langage de commande de l'environnement de programmation. ✓ Connaissance approfondie des constructions de base des langages C et C ++: types de données, instructions, fonctions prédéfinies et utilisateur, techniques de programmation, etc. ✓ Capacité de comprendre les codes d'erreur du compilateur et du système d'exploitation. ✓ Capacité de formuler des modèles mathématiques de problèmes résolus. ✓ Possibilité d'utiliser des langages de programmation et des environnements plus efficaces.

Compétences transversales	CT1. Appliquer les principes, les normes et les valeurs de l'éthique professionnelle.
	CT2. Identifier, décrire et gérer les activités organisées en équipe ; développement des capacités de communication et de collaboration, ainsi que d'assumer les différents rôles (exécution et leadership)
	CT3. Faire preuve de l'esprit d'initiative et d'action pour mettre à jour les connaissances professionnelles, économique et de la culture organisationnelle

6. Objectifs de l'unité de cours / module

Objectif général	**Assurer une formation théorique et pratique en conception et programmation orientées objet (en C++): comprendre les apports de l'approche orientée objet au domaine du génie logiciel, appliquer les concepts des technologies orientées objet en C++.**
Objectifs spécifiques	**Comprendre les principes fondamentaux de l'approche orientée objet; identifier les composants et savoir comment mettre en œuvre un modèle orienté objet avec C++.**

7. Contenu de l'unité de cours / module

Thématique des activités didactiques	Nombre d'heures	
	enseignement à temps plein	enseignement à temps partiel
Thème des cours		
T1. Introduction à la POO. Différences entre les langages de programmation C et C ++.	2	
T2. Classe - mécanisme d'abstraction. Définition de types abstraits utilisant des structures et des classes. Structure et classe: analyse comparative. Visibilité et accès aux éléments de classe. Encapsulation - le mécanisme de l'unification des données et des fonctions. Deux visions de la classe: le côté de l'interface et la partie d'implémentation. Définition des fonctions membres, des fonctions d'accès (membres, amies) aux variables privées de la classe.	2	
T3. Constructeurs - fonctions d'initialisation des objets. Initialisation des objets et des vecteurs d'objets. Types de constructeurs. Constructeur de recopie - la nécessité de le définir. Destructeur. Le lien entre les opérateurs *new* et *delete* effectués par les constructeurs et le destructeur. Le pointeur *this* et son utilisation. Les composants statiques de la classe. Motiver l'utilisation d'éléments statiques. Variables statiques, méthodes statiques.	2	
T4. Fonctions et classes amies (friend). Motivation et définition de type *friend.*	2	
T5. Surcharge des opérateurs - définition des opérations sur des objets. Motivation de la surcharge. Classement des opérateurs (unaires, binaires, préfixes et postfixes). La syntaxe de la définition de l'opérateur. Formes de surcharge / redéfinition (en tant que fonctions membres et fonctions amies). Dépassement de la récupération de données et lecture des opérateurs (<<, >>). Liste des opérateurs qui ne peuvent pas être redéfinis. Redéfinir les opérateurs [], () et ->.	2	

T6. Héritage - mécanisme de réutilisation du code. Formes d'héritage (public, protégé et privé). Classe de base et classe dérivée. Composants de classe protégés. Redéfinir les méthodes héritées. Héritage et constructeurs, destructeur (appel du constructeur de la classe de base). Hiérarchie - arbre de dérivation.	2	
T7. Héritage multiple et ses problèmes. Héritage virtuel.	2	
T8. Composition, agrégation et association. La différence entre la relation de composition et d'agrégation. Réutilisez le code dans une autre classe en définissant une référence à un objet dans la classe appropriée. L'appel des constructeurs pour les objets inclus dans une autre classe.	2	
T9. Polymorphisme. Fonctions virtuelles, types de données dynamiques. Variables polymorphiques.	2	
T10. Fonctions virtuelles pures - classes abstraites.	2	
T11. Namespace. Définir les espaces de noms, la nécessité de les définir.	2	
T12. Template. Modèles de fonctions et classes. Spécialisations.	2	
T13. Exceptions. Traiter les exceptions en C ++ en utilisant *try, throw* et *catch*.	2	
T14. Bibliothèque de modèles standard (STL - Standard Template Library). Types de conteneurs STL: conteneurs de séquence, conteneurs associatifs et adaptateurs de conteneur.	4	
Total des cours:	30	

Thématique des activités didactiques	Nombre d'heures	
	enseignement à temps plein	enseignement à temps partiel
Thèmes des travaux pratiques		
TP1 Programmation par abstraction de données. Définition d'une structure, opérations sur les structures dans le langage C	4	
TP2 Définir une classe, définir les constructeurs, le destructeur et les fonctions d'accès à la partie privée de la classe.	4	
TP3 Surcharge des opérateurs. Fonctions membres et fonctions amies.	4	
TP4 Héritage, composition et agrégation.	4	
TP5 Héritage multiple. Héritage virtuel.	4	
TP6 Polymorphisme. Fonctions virtuelles. Classes abstraites.	4	
TP7 Modèles des fonctions et des classes (template). Spécialisations.	6	
Total des travaux pratiques:	30	

Thématique des activités didactiques	Nombre d'heures	
	enseignement à temps plein	enseignement à temps partiel
Thèmes des travaux dirigés :		
TD1 Différences entre les langages de programmation C et C ++. Définissez le même type de données abstraites en utilisant la structure et la classe.	2	
TD2 Définir une classe. Constructeurs, destructeurs, fonctions membres. Fonctions et classes amies (friend).	2	
TD3 Redéfinir des opérateurs pour les opérations sur les objets. Choisir la forme de surcharge (fonctions membres et fonctions amies) des opérateurs mathématiques, de comparaison, etc. Héritage. Syntaxe de la définition d'une classe dérivée. Héritage simple, hiérarchique.	2	

TD4 Héritage multiple. Héritage virtuel. Relations de composition, d'agrégation et d'association.	2	
TD5 Polymorphisme. Fonctionnalités virtuelles, fonctions virtuelles pures, types de données dynamiques. Variables polymorphes, classes abstraites.	2	
TD6 Déclaration d'espace de noms (namespace), directive *using*. Déclaration des modèles ((template) de fonction, classe.	2	
TD7 Traitement des exceptions en C ++. Utilisation des conteneurs STL dans les programmes.	3	
Total des travaux dirigés:	15	

8. Références bibliographiques

Principales	1. Bjarne Stroustrup, The C++ Programming Language, 4rd ed., Addison-Wesley Professional, 1368 p, 2013 2. Harry. H. Chaudhary, Effective C++ : Easy Beginner's To Experts Edition, Createspace LLC USA, 260 p., 2014 3. Scott Meyers, Effective C++: 55 Specific Ways to Improve Your Programs and Designs, Addison-Wesley, 4rd ed., 297 p., 2011 4. D. Ryan Stephens, Christopher Diggins, Jonathan Turkanis, Jeff Cogswell, C++ Cookbook: Solutions and Examples for C++ Programmers, O'Reilly Media, 573 p. 2006.
Supplémentaires	5. Bjarne Stroustrup, Programming: Principles and Practice Using C++, 2nd ed., Addison-Wesley Professional, 1272 p, 2014 6. Margaret A. Ellis, Bjarne Straustrup, The Annotated C++ Reference Manual, Addison-Wesley, 480 p., 1990 7. Nicoleta Liviana Tudor, Bazele programării în limbajul C++, Editura MATRIX ROM, Bucuresti, 79 p., 2010.

Evaluation

Périodique		Actuelle	Étude individuelle	Projet / thèse	Examen
EP 1	EP 2				
15%	15%	15%	15%	-	40%

Normes de rendement minimum

Présence et activité aux travaux pratiques

Obtenir le score minimal de "5" pour chacune des évaluations périodiques et des travaux pratiques ; Démonstration de l'assimilation des informations fournies pendant le cours et des compétences pour dessiner les diagrammes nécessaires à la conception d'un produit de programme à l'examen final