Cégep de Sainte-Foy	**Programme des Techniques de l'informatique** **Programmation de jeux vidéo** Département des Techniques d'intégration multimédia	Automne **2017**

Algorithmes et structures de données II

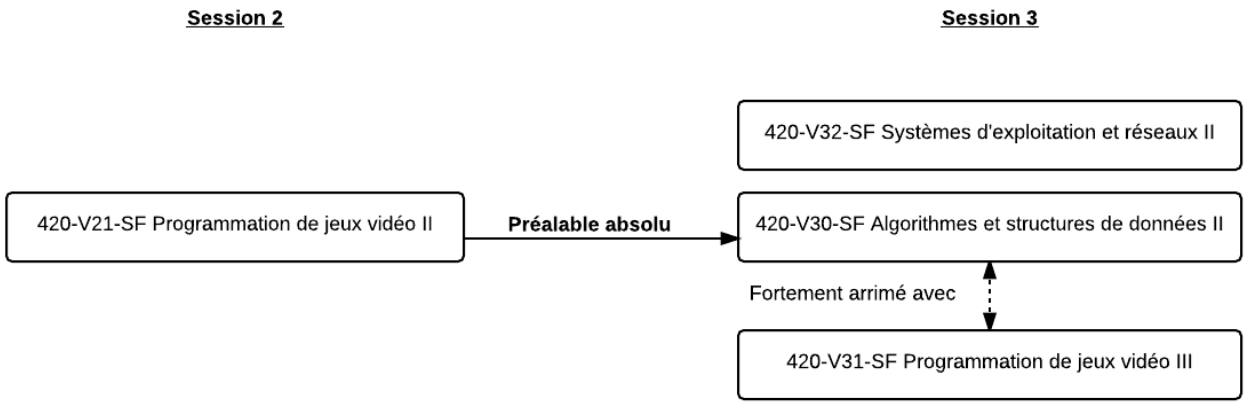
420-V30-SF	Pondération 2-2-2	A Pourvoir, local C-###, casier ### 418-659-6600 #----, apourvoir@cegep-ste-foy.qc.ca

Présentation générale du cours

Le cours « Algorithmes et structures de données II» (420-V30-SF) destiné à la clientèle étudiante du programme « Techniques de l'informatique – Programmation de jeux vidéo » a pour objectif d'aborder les notions relatives à la représentation et à l'exploitation de données dans la mémoire de l'ordinateur. On y aborde les mécanismes d'allocation et de libération de mémoire dynamique à l'aide du langage C++. L'étudiant apprendra les notions algorithmiques et celles reliées à la programmation de structures de données dynamiques (par exemple, piles, files, etc.) et les appliquera dans des contextes reliés aux jeux vidéo. L'étudiant apprendra finalement les techniques fondamentales reliées au débogage de programmes informatiques.

Ce cours permet donc à l'étudiant de consolider ses bases en programmation structurée en plus d'utiliser les ressources disponibles pour solutionner les problématiques qui lui seront soumises, augmentant ainsi son autonomie.

Situation du cours dans le programme



Le préalable pour ce cours est le cours « 420-V21-SF Programmation de jeux vidéo II ».

Ce cours est donné en parallèle et fortement arrimé avec le cours « 420-V31-SF Programmation de jeux vidéo III ». Il est également donné en parallèle avec le cours « 420-V32-SF Systèmes d'exploitation et réseaux II». Il n'est préalable à aucun autre cours du programme.

Objectifs spécifiques du cours

Le cours vise spécifiquement à développer les éléments de compétences suivants :

Énoncé(s) de la compétence	Éléments de la compétence
016S - Exploiter un langage de programmation structurée	1 Préparer l'environnement de programmation
	2 Adapter l'algorithme aux contraintes du langage de programmation.
	3 Traduire l'algorithme dans le langage de programmation.
	4 Compiler le programme
	5 Valider le programme
016W – Produire des algorithmes	1 Analyser la situation.
	2 Mettre au point l'algorithme.
	3 Valider l'algorithme.
0170 – Organiser et exploiter des données	1 Procéder à l'organisation logique des données sur les supports physiques.
	2 Procéder à l'organisation logique des données en mémoire.
	4 Exploiter des données en mémoire.
0171 – Corriger des programmes	1 Analyser le problème
	2 Déterminer la nature du problème
	3 Corriger le problème
	4 Valider la solution
	5 Apporter les corrections à la documentation

Stratégies pédagogiques et modalités de participation

Ce cours se déroule sous forme d'exposés magistraux, de démonstrations et d'ateliers en laboratoire pour un total de 60 heures en classe auxquelles s'ajoutent environ 30 heures de travail personnel. Le professeur intervient pour introduire et commenter les concepts et les notions nécessaires à la réalisation de chaque programme. Il agit régulièrement comme un guide accompagnant l'étudiant dans son cheminement.

Des exercices fréquents permettront à l'étudiant d'appliquer les méthodes et techniques enseignées. Les travaux pratiques seront directement en lien avec les jeux vidéo.

On développera l'autonomie et l'auto apprentissage des étudiants en insistant sur l'exploration de l'environnement de travail ainsi que sur l'utilisation des ressources documentaires disponibles.

Stratégies d'évaluation

Évaluation formative

Tout au long de la session, de nombreux exercices et laboratoires formatifs seront réalisés par l'étudiante ou l'étudiant. Certains de ces exercices et laboratoires devront être remis par chaque étudiant et seront ensuite corrigés en classe. Ces exercices et laboratoires viseront la compréhension et l'application de la matière, de même que la préparation aux évaluations sommatives. L'étudiante ou l'étudiant aura également l'occasion de réaliser au moins un ou des quiz préparatoires par examen.

Pour motiver les étudiants à remettre les exercices et laboratoires formatifs ainsi qu'à se préparer aux quiz formatifs, **10% de la pondération du cours sera réservée à l'évaluation formative.**

Évaluation sommative

L'évaluation des compétences se fera à l'aide de trois examens sur des notions théoriques pour 60% et de trois travaux pratiques pour 30%.

Nature des évaluations	Pondération	Échéance	Contexte de réalisation
Travail Pratique 1	10%	Semaine 5	Individuel
Travail Pratique 2	10%	Semaine 11	Individuel
Travail Pratique 3	20%	Semaine 15	En équipe
Examen 1	20%	Semaine 7	Écrit, individuel
Examen Synthèse	30%	Semaine 15	Écrit, individuel

Note de passage

Pour obtenir la note de passage d'un cours, l'étudiant doit obtenir **une note finale minimale de 60%.**

Médiagraphie & environnement de développement

1. Sites internet
    - Tutoriaux sur le langage C++
        - https://openclassrooms.com/courses/programmez-avec-le-langage-c
        - https://msdn.microsoft.com/en-us/library/aa733747(v=vs.60).aspx
        - https://fr.wikibooks.org/wiki/Programmation_C%2B%2B
    - Algorithmique
        - http://www.pise.info/algo/codage.htm
        - http://fr.m.wikibooks.org/wiki/Algorithmique_imp%C3%A9rative

Environnement de développement obligatoire

- Microsoft Visual Studio Ultimate 2013 (disponible gratuitement via MSDNAA)

Matériel obligatoire

Outre l'ordinateur portable, aucun autre matériel n'est obligatoire pour ce cours. Cependant, une option de stockage supplémentaire (comme par exemple une clé USB) est fortement recommandée.

Calendrier des blocs et des remises

Le calendrier suivant est donné à titre indicatif seulement et peut être modifié en fonction des circonstances. En ce qui concerne les dates des examens et celles de remise des travaux, il va de soi que les étudiants seront avisés conformément à ce que prévoit la Politique d'évaluation des apprentissages du Collège.

Bloc 1 – Algorithmique avancée et allocation dynamique

Bloc 2 – Structures de données dynamiques

Énoncé de la compétence	Éléments de la compétence
016S - Exploiter un langage de programmation structurée.	016S-1 Préparer l'environnement de programmation 016S-2 Adapter l'algorithme aux contraintes du langage de programmation. 016S-3 Traduire l'algorithme dans le langage de programmation. 016S-4 Compiler le programme 016S-5 Valider la solution
016W - Produire des algorithmes	016W-1 Analyser la situation. 016W-2 Mettre au point l'algorithme. 016W-3 Valider l'algorithme.
0170 - Organiser et exploiter des données	0170-1 Procéder à l'organisation logique des données sur les supports physiques 0170-2 Procéder à l'organisation logique des données en mémoire 0170-4 Exploiter des données en mémoire.
0171 - Corriger des programmes	0171-1 Analyser le problème 0171-2 Déterminer la nature du problème 0171-3 Corriger le problème 0171-4 Valider la solution 0171-5 Apporter les corrections à la documentation

	Critères de performance	Habiletés	Contenus
016S-1	1.1 Vérification méthodique de l'accès aux différents éléments physiques et logiques de l'environnement. 1.2 Configuration de l'environnement appropriée aux caractéristiques de la situation. 1.3 Personnalisation de l'environnement efficace et conforme aux exigences de l'entreprise.	1.1 Vérifier l'accès aux éléments physiques et logiques (ex : base de données, gestionnaires de codes sources) 1.2 Installer et configurer l'environnement de développement intégré (IDE). 1.3 Choisir et installer les modules d'extension de l'environnement de développement intégré (IDE) utiles au développement du programme. 1.3 Choisir et installer les bibliothèques de code nécessaires au développement du programme. 1.3 Configurer les connexions aux différents services (par exemple, base de données, gestionnaires de codes sources).	Principales possibilités offertes par l'environnement de développement intégré (IDE). Principaux modules d'extension de l'environnement de développement intégré (IDE) et leur installation Bibliothèques de code et leur installation Connexion à différents services

016S-2	2.1 Modification appropriée de la représentation des données. 2.2 Adaptation correcte des conditions d'exécution. 2.3 Modification appropriée des structures de traitement. 2.4 Adaptation appropriée de la séquence des opérations.	Cet élément de compétence est notamment lié à certains vieux langages de programmation (ex : Cobol). Il a été jugé qu'il est implicitement inclut dans l'élément de compétence 3.	
016S-3	3.1 Utilisation efficace des fonctionnalités d'édition de l'environnement. 3.2 Application des règles de syntaxe et de sémantique propres au langage utilisé. 3.3 Application rigoureuse des standards de codification. 3.4 Application judicieuse des principes de la programmation structurée. 3.5 Mise à profit judicieuse des possibilités du langage. 3.6 Consignation des commentaires pertinents et conformes aux exigences de l'entreprise.	3.1 3.2 3.3 3.4 Programmer l'algorithme dans le langage de programmation choisi en utilisant les principes de programmation structurée. 3.5 Comprendre et utiliser les différentes fonctionnalités du langage de programmation choisi. 3.6 Programmer l'algorithme selon les normes de codification. 3.6 Commenter le code de façon pertinente.	Syntaxe et sémantique du langage retenu Principes de programmation structurée (par exemple, structures de contrôle, découpage en fonctions) Aide en ligne Principales classes et méthodes disponibles dans le langage retenu Normes de programmation Documentation du programme
016S-4	4.1 Utilisation efficace des fonctionnalités de compilation de l'environnement. 4.2 Repérage des erreurs de compilation. 4.3 Correction des erreurs de compilation.	4.1 Utiliser les options de génération (debug ou release) selon le contexte. 4.2 Distinguer les principales étapes de génération. 4.2 Distinguer et comprendre les erreurs de génération de façon à pouvoir les corriger. 4.3 Corriger les erreurs de génération.	Options de génération Étapes de génération Différents types d'erreurs Aide en ligne
016S-5	5.2 Préparation correcte des jeux d'essai nécessaires à la vérification du fonctionnement du programme. 5.3 Interprétation juste des résultats. 5.4 Débogage approprié du programme selon l'algorithme.	Les tests d'intégration ne devraient pas être enseignés dans le cadre de cette compétence, mais les étudiants devraient connaître leur existence et les distinguer des tests unitaires au moment d'apprendre les tests unitaires. Distinguer les tests de fiabilité des tests de robustesse. 5.1 5.3 Exécuter les tests unitaires et interpréter les résultats. 5.1 5.4 Choisir et utiliser les fonctionnalités du débogueur les plus pertinentes au contexte.	Tests unitaires de fiabilité Tests unitaires de robustesse
016W-1	1.1 Établissement correct des données d'entrée. 1.2 Établissement correct des données de sortie. 1.3 Établissement correct de la nature des traitements. 1.4 Détermination correct des conditions d'exécution de l'algorithme.	1.1 1.2 Identifier les données requises et leur type. 1.3 Identifier la nature des traitements requis. 1.4 Préparer un jeu d'essais approprié. 1.4 Identifier les conditions d'exécution de l'algorithme.	Structures de contrôle (séquences, conditionnelles, boucles) Structures de données (constantes, variables, tableaux) Jeux d'essai Choix des valeurs à tester (par exemple, cas généraux, cas limites et cas particuliers)

016W-2	2.1 Choix d'un mode de représentation de l'algorithme conforme aux exigences de l'entreprise. 2.2 Détermination d'une séquence logique des opérations. 2.3 Détermination des structures de traitement appropriées à chacune des opérations. 2.4 Application rigoureuse des règles de syntaxe propres au mode de représentation retenu. 2.5 Recherche d'une solution algorithmique efficace. 2.6 Représentation précise de la solution algorithmique retenue. 2.7 Présence de toute l'information nécessaire à l'interprétation de l'algorithme.	2.1 Utiliser différents modes de représentation d'algorithmes. 2.2 Déterminer la séquence des opérations nécessaires aux résultats à produire. 2.3 Déterminer les structures de traitement nécessaires aux résultats à produire. 2.6 Appliquer les techniques de représentation des algorithmes selon le modèle retenu. Commenter des algorithmes.	Modes de représentation schématique et textuel Expressions (par exemples, arithmétiques, logiques, d'affection) Fonctions et paramètres Trace d'algorithmes Emplacement et nature des commentaires
016W-3	3.1 Vérification de la pertinence de la solution compte tenu de la situation initiale. 3.2 Détermination des erreurs et des lacunes de la solution algorithmique mise au point. 3.3 Modification appropriée de la solution algorithmique.	3.1 Identifier les forces et les faiblesses d'une solution algorithmique afin d'en déterminer la pertinence. 3.2 Identifier les failles d'une solution algorithmique. 3.3 Appliquer la série de tests énoncés dans un jeu d'essais. 3.3 Apporter les correctifs identifiés pour rendre une solution conforme aux exigences initiales. 3.3 Produire la documentation nécessaire au suivi des modifications.	Critères de validation d'algorithmes (simplicité, exactitude)
0170-1	1.1 Analyse du contexte d'utilisation des données. 1.2 Comparaison des possibilités offertes par les différents types de supports physiques. 1.3 Détermination des supports physiques appropriés au contexte. 1.4 Détermination judicieuse du mode d'accès aux fichiers.	1.1 Déterminer le contexte d'utilisation des données. 1.2 Si l'organisation des données est possible sur plusieurs types de supports physiques, comparer les possibilités offertes par les différents types de supports physiques. 1.3 Choisir les supports physiques appropriés au contexte et leur mode d'accès. 1.4 Il a été jugé que ce critère est désuet.	Types de supports physiques et leurs caractéristiques Types de structures de stockage de données

0170-2	2.1 Analyse du contexte d'utilisation des données. 2.2 Comparaison des possibilités offertes par les différentes structures de données. 2.3 Choix des structures de données appropriées au contexte. 2.4 Création correcte de piles, files, listes, arbres et tableaux dans un langage de programmation. 2.5 Choix approprié du mode d'allocation en mémoire des structures de données.	2.1 Déterminer le contexte d'utilisation des données. 2.2 Si l'organisation des données est possible avec plusieurs structures de données, comparer les possibilités offertes par les différents structures de données. 2.3 Choisir les structures de données appropriées au contexte. 2.4 Programmer des structures de données. 2.4 Adapter des structures de données selon les besoins. 2.5 Choisir le mode d'allocation (statique ou dynamique) approprié au contexte.	Différentes structures de données et leurs caractéristiques Gestion de la mémoire dans un programme Organisation interne de la mémoire vive dans un programme. Programmation de structures de données Caractéristiques des principales classes collections Modes d'allocation statique et dynamique
0170-4	4.1 Mise à jour correcte des structures de données en fonction des opérations. 4.2 Mise à jour correcte des données. 4.3 Utilisation appropriée de la mémoire en fonction du mode d'allocation retenu. 4.4 Application des techniques de réorganisation des données en mémoire.	4.1 4.2 Programmer l'accès et la mise à jour des données et des structures de données. 4.3 Gérer l'allocation de la mémoire pour les données. 4.4 Réorganiser des données en mémoire selon les besoins.	Instructions d'accès et de mise à jour des données et des structures de données en mémoire Principaux mécanismes de gestion de la mémoire Algorithmes de tri
0171-1	1.1 Reconstitution du problème dans l'environnement approprié et dans les conditions originales. 1.2 Formulation méthodique d'hypothèses visant à trouver le problème. 1.3 Utilisation appropriée des outils de débogage en vue de vérifier les hypothèses. 1.4 Détermination de la composante du programme où se situe le problème.	1.1 Identifier une séquence d'opérations permettant de reproduire le problème. 1.2 Développer des stratégies de résolution de problèmes. 1.3 Utiliser des outils et techniques d'aide au débogage. 1.4 Déterminer la composante à corriger.	Étapes de reproduction Approche systématique pour la résolution de problèmes complexes Base de données de bogues Outils et techniques de débogage
0171-2	2.1 Examen des documents pertinents associés à la composante du programme en cause. 2.2 Établissement d'hypothèses pertinentes sur la nature du problème. 2.3 Choix judicieux des tests visant à vérifier les hypothèses. 2.4 Utilisation appropriée des outils de débogage. 2.5 Interprétation correcte des résultats. 2.6 Déduction de la nature exacte du problème.	2.1 Comprendre la documentation existante. 2.2 Établir des hypothèses sur la nature du problème. 2.3 Développer des données de tests aptes à valider les hypothèses. 2.3 2.5 Réaliser des tests pertinents et interpréter les résultats. 2.4 Utiliser des outils et techniques d'aide au débogage. 2.6 Conclure sur la nature du problème.	Outils et techniques de débogage

0171-3	3.1 Résolution efficace des problèmes de conception de l'algorithme. 3.2 Résolution efficace des problèmes de traduction de l'algorithme dans le langage de programmation. 3.3 Résolution efficace des problèmes d'utilisation du langage de programmation.	3.1 Résoudre des problèmes de conception d'algorithme. 3.2 Résoudre des problèmes de traduction d'algorithme. 3.3 Résoudre des problèmes d'utilisation du langage de programmation.	Contenus des compétences 016W, 016S et 016T
0171-4	4.1 Analyse de l'effet de la solution sur l'ensemble du programme. 4.2 Préparation des jeux d'essai appropriés. 4.3 Interprétation juste des résultats. 4.4 Fonctionnement correct du programme. 4.5 Présentation de la solution à la supérieure ou au supérieur pour approbation.	4.1 4.2 4.3 Connaître les éléments d'un plan de tests. 4.2 4.3 4.4 Développer et réaliser des jeux d'essai permettant de confirmer le fonctionnement du programme comme prévu. 4.5 Obtenir l'approbation du supérieur ou de la supérieure pour la solution	Jeux d'essai Tests automatisés
0171-5	5.1 Consignation minutieuse de toute l'information liée au traitement du problème.	5.1 Mettre à jour la documentation afin qu'elle reflète les corrections apportées au programme. 5.1 Utilisation adéquate d'un système de gestion des sources.	Commentaires en lien avec l'utilisation d'un système de gestion des sources

Contexte d'apprentissage	Activités d'apprentissage
• Individuellement. • Exercices formatifs régulièrement À partir - de concepts de programmation - de cas concret d'utilisation À l'aide - d'un ordinateur	**Exercices** : Formatifs tout au long du bloc 1 et du bloc 2 Bloc 1 : Résolution de problèmes, fichiers, algorithmique, allocation dynamique, vecteurs. Bloc 2 : notions de C++, structures de données (pile, liste, file), classes génériques. Quiz formatifs préparatoires à l'examen du bloc 1 et à l'examen synthèse à la suite du bloc 2 **Évaluations sommatives :** Travaux pratiques en lien avec les jeux vidéo Examens écrits

Calendrier détaillé

Semaines	Heures consacrées	Activités pédagogiques	Évaluations	Blocs du cours
1 21/08/17 (4 h)	1	Présentation du cours	s/o	Bloc 1
	3	Techniques de résolution de problèmes en programmation	Formative	Bloc 1
2 28/08/17 (4 h)	2	Efficacité d'algorithmes et approches algorithmiques classiques	Formative	Bloc 1
	2	Allocation dynamique en C++	Formative	Bloc 1
3 04/09/17 (4 h)	2	Allocation dynamique en C++ (suite)	Formative	Bloc 1
	2	Composition et diagrammes de classe	Formative	Bloc 1
4 11/09/17 (4 h)	1	Quiz formatif	Formative	Bloc 1
	5	**Travail Pratique #1**	**Sommative 10%**	**Bloc 1**
5 18/09/17 (4 h)	5	Les vecteurs	Formative	Bloc 1
6 25/09/17 (4 h)	1	Quiz Formatif	Formative	Bloc 1
7 02/10/17 (4 h)	2	Retour sur le Travail Pratique #1 et révision pour l'examen 1	Formative	Bloc 1
	2	**Examen 1**	**Sommative 20%**	**Bloc 1**
10/10/16	Semaine de relâche			

8 16/10/17 (4 h)	1	Retour sur l'examen 1	Formative	Bloc 1
	3	Complément de formation en C++	Formative	Bloc 2
9 23/10/17 (4 h)	4	Les piles	Formative	Bloc 2
10 30/10/17 (4 h)	2	Les files	Formative	Bloc 2
	1	Introduction à la gestion des fichiers sources	Formative	Bloc2
11 06/11/17 (4 h)	**5**	**Travail Pratique #2**	**Sommative 10%**	**Bloc 2**
12 13/11/17 (4 h)	4	Les listes	Formative	Bloc 2
13 20/11/17 (4 h)	3	Retour sur le Travail Pratique #2 Les classes génériques en C++	Formative	Bloc 2
	1	Quiz formatif	Formative	Bloc 2
14 27/11/17 (4 h)	**5**	**Travail Pratique #3**	**Sommative 20%**	**Bloc 2**
15 04/12/17 (4 h)	1	Révision examen final	Formative	Bloc 2
	2	**Examen Synthèse**	**Sommative 30%**	**Synthèse**

Règles départementales

Voir le document sur les règles départementales remis avec ce plan de cours.