Séance 1 : autour d’une file d’attente (*Queue*)

Les objectifs de cette séance, et du travail individuel à fournir d’ici la séance suivante, sont :

1-Se remettre à flot avec les **concepts de base du C** vus au précédent quadrimestre : types, variables, expressions arithmétiques, conditions, boucles, chaines de caractères, fonctions, structures, pointeurs

2-Comprendre et savoir utiliser l’**allocation dynamique de mémoire en C** (malloc) et la libération de la mémoire allouée (free)

3-Comprendre la différence entre les **fichiers .c et les fichier d’entête .h** et être capables, lors de l’écriture d’un programme, de jongler entre les deux

4-**Compiler** des projets composés de **plusieurs fichiers** .c et .h

5-Comprendre le principe d’un **type abstrait de données** (abstract data type, ADT)

6-Être capable d’**implémenter concrètement un ADT** donné, de plusieurs manières

7-Utiliser les ADT "**file d’attente**" et "**pile**" pour résoudre des problèmes concrets en C

1.Type Queue

Une file d’attente, ou "queue" en anglais, est un type de donnée où les données sont récupérées selon un mécanisme "first in first out".

1.1.Concret ou abstrait ?

Jetez un œil sur le fichier queue.h reproduit ici. S’agit-il d’un type de donnée abstrait ou concret ? Justifiez vos réponses en vous basant sur les définitions de ces types de données.

1.2.Utilisation de la Queue

Sans connaître l’implémentation de la Queue, pouvez-vous prédire ce qui va être affiché lors de l’exécution du programme "manipulation_queue.c" ? Si oui, indiquez ce qui va être affiché.

Est-ce que plusieurs implémentations concrètes correctes de la Queue sont possibles ? Cela va-t-il influencer l’expérience de l’utilisateur du programme "manipulation_queue.c" ? Si oui, en quoi ? Sinon, pourquoi ?

1.3. Définition d’une Queue

Identifiez les éléments clés qui permettent de définir une Queue (en particulier : les opérations et propriétés d’une Queue).

2.Implémentation de la Queue

Vous allez implémenter une Queue à partir du fichier d’entête qui a été donné. Suivez les étapes !

2.1. Coffre-fort

Examinez le code "coffre_fort.c" qui vous est fourni et utilisez-le pour clarifier les éléments suivants :

-Comment est définie concrètement une structure en C ? En particulier, comment la nommer, comment créer un type synonyme (ou alias), comment définir des champs ?

-Comment créer, en C, un tableau dont on ne connaît pas la taille avant l’exécution du programme ?

-À quoi sert "malloc" ? Comment l’employer ? Que prend-il comme argument ? Que renvoie-t-il ?

-À quoi sert "free" ? Pourquoi l’employer ? Que prend-il en argument ?

2.2.Listes simplement chainées

La liste simplement chainée est un type de donné concret dynamique (dont la taille n’est pas fixée à l’avance).

Déterminez ce qu’est concrètement une liste simplement chainée et proposez-en une implémentation en C.

Suggestion : définissez deux structures. Une pour la liste "en elle-même" et une pour les nœuds de la liste, qui s’enchainent dont le dernier pointe vers NULL.

Proposez ensuite une implémentation d’une liste doublement chainée et d’une liste doublement chainée circulaire.

2.3. Tests de la Queue

Écrivez une fonction qui vous permettra de tester la Queue que vous allez implémenter à l’étape suivante. Dans cette fonction, vous veillerez à tester chacune des 5 fonctions définie dans le fichier d’entête. N’hésitez pas à vous répartir le travail, pour que chacun et chacune se concentre sur une ou deux fonction et prévoie de la tester à fond. Veillez à ce que cette fonction indique à chaque fois quelque chose du type:

Test de la fonction f1

Test 1 : réussi

Test 2 : réussi

Test 3 : échoué

…

Test de la fonction f2

Test 1 : réussi

Test 2 : échoué

…

2.4.Implémentation de la Queue

Proposez deux manières d’implémenter la Queue, au moyen d’une structure statique (par exemple, un tableau) et au moyen d’une structure dynamique (par exemple, une liste chainée simple, doublement chainée, chainée circulaire, etc.).

Déterminez les avantages et inconvénients de chacune de ces deux implémentations.

3.La pile (Stack)

3.1.C’est quoi une pile ?

Stack est un ADT. Déterminez les opérations et propriétés qui permettent de le définir.

3.2.Fichier d’entête

À partir de la définition que vous avez donnée d’une pile, déterminez un fichier d’entête (.h) qui contient la signature des fonctions qui permettent de définir cet ADT. Pour chacune de ces fonctions, indiquez bien ses préconditions et postconditions.

3.3.Implémentation d’une pile

Implémentez la pile définie précédemment dans un fichier .c de deux façons différentes : au moyen de tableaux et au moyen de listes chainées (simplement, ou doublement, ou circulaire).

4.Des exercices supplémentaires

Sur Webcampus, vous trouverez 4 exercices supplémentaires à réaliser qui vous permettront de travailler les différents concepts vus lors de cette séance :

-Un exercice de compression de données, pour manipuler des tableaux et des files

-Un exercice de chiffrement de données, pour manipuler des files

-Un exercice de calculatrice, pour manipuler des piles

-Un exercice de bataille, pour manipuler des piles