Séance 2 : autour d’une file de priorité

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

8-Être capable de c**omprendre un ADT fourni dans une notation formelle**

9-Créer et manipuler **des fonctions qui prennent en argument d’autres fonctions**

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

11-Être capable d’écrire une **suite de tests simples** pour tester l’implémentation d’un ADT.

12-Utiliser l’ADT "**file de priorité**" pour résoudre des problèmes concrets en C

1. File de priorité

Une file de priorité, ou "priority queue" en anglais, est un type de donnée où les éléments sont récupérés selon un mécanisme dans leur ordre de priorité en fonction d’un **rang** qui leur est assigné.

1.1.ADT File de Priorité

Dans un article de 1990[1], l’informaticienne Nell Dale, définit l’ADT "File de Priorité" de cette façon :

Notation:

PQ’ the priority queue on entry to the operation
PQ the priority queue on exit from the operation
⊥ symbol for undefined
Max maximum number of items in PQ
Item <e, p_e> in ItemType where
e in ElementType (the type of objects being put on the queue)
p_e in PriorityType, and
p_e is the priority of e

Operations:

Create(VAR PQ : PQType)
Pre: True
Post: PQ <> ⊥ AND |PQ| = 0

Enque(VAR PQ : PQType; Item : ItemType)
Pre: PQ’ <> ⊥ AND |PQ’| <> Max
Post: PQ = PQ’ U {Item}

Serve(VAR PQ : PQType; VAR Item : ItemType)
Pre: PQ’ <> ⊥ AND |PQ’| <> 0
Post: PQ = PQ’ - {<e, p_e> | ∀<x, p_x> in PQ’, p_e >= p_x }

Empty(PQ : PQType) : Boolean
Pre: PQ’ <> ⊥
Post: Empty = (|PQ’| = 0)

Full(PQ : PQType) : Boolean
Pre: PQ’ <> ⊥
Post: Full = (|PQ’| = Max)

Fournissez un fichier entête (.h) correspondant à cet ADT.
Notez que dans cet ADT, il n’y a pas de return mais des variables de sorties passées en argument. N’employez pas ce type de variable de sortie.

Basez-vous sur PQ.h et complétez-le.

Ajoutez à la notation formelle une opération "peek", qui renvoie l’élément avec la plus haute priorité mais ne modifie pas la file de priorité. Ajoutez-la ensuite à votre fichier entête.

1.2.Passer une fonction en argument

Dans PQ.h, vous remarquez que la fonction create a un argument étrange. Il s’agit d’une fonction passée en argument qui permettra de définir une relation d’ordre dans notre file de priorité, en disant, pour deux éléments donnés, si le premier est plus petit que le deuxième. Ça permet de définir un ordre.

---

1. Dale, Nell. "If you were lost on a desert island, what one ADT would you like to have with you?." *ACM SIGCSE Bulletin* 22.1 (1990): 139-142.

Pour comprendre le fonctionnement du passage de fonction en argument, complétez le programme C printmap.c qui comportera les fonctions suivantes :

-Une fonction doubler, qui prend en argument un entier et renvoie cet entier multiplié par deux.

-Une fonction inc, qui prend en argument un entier et renvoie cet entier incrémenté de 1.

-Une fonction printmap qui ne renvoie rien, qui prend en argument un tableau d'entiers, sa taille et une fonction qui prend en argument un entier et renvoie un entier et qui modifie le tableau passé en argument en appliquant à chacun de ses éléments la fonction passée en argument, et l'affiche.

1.3. Testez votre file de priorité

Déterminer une série de tests que devront passer votre file de priorité, sous la forme d'une fonction de test qui crée une file de priorité, lui applique une série d'opération et fait un certain nombre de vérifications. Veillez à ce que vos tests soient le plus complet possible (testez toutes les fonctions, testez des cas standards et des cas particuliers, etc.). Cette fonction, idéalement, affiche, test par test, s'il est réussi ou pas. Ça vous permettra, lorsque la file de priorité sera implémentée, de la tester.

1.4. Implémentez une file de priorité avec un type déjà vu

Implémentez PQ au moyen d'un type que vous avez déjà employé ou défini précédemment : tableau, liste chainée, file, pile. Trouvez ensuite quelqu'un qui l'a implémenté au moyen d'un autre type de données et évaluez, fonction par fonction, la différence de complexité temporelle et spatiale.

1.5. Implémenter une file de priorité avec un Heap / Tas binaire

Un tas binaire (heap) est une manière intéressante d'implémenter une file de priorité comme d'un arbre binaire presque complet ordonné (chaque nœud parent est plus grand ou égal à son nœud enfant), au moyen par exemple d'un tableau (dont le premier élément vaut la racine, puis les deux éléments suivant le niveau 1 de l'arbre, puis les 4 éléments suivants le niveau 2 et ainsi de suite).

Implémentez la file de priorité définie précédemment comme un tas binaire. N'hésitez pas à consulter la vidéo suivante qui explique visuellement comment ajouter et retirer des éléments :

File de priorité et tas binaire , François Schwarzentruber

Compare l'efficacité de cette façon d'implémenter la file de priorité avec ton implémentation précédente. Que constates-tu ?

2.Programme de suggestion de livres

Écris un programme C qui permette à un utilisateur, une fois lancé, d'encoder des livres (titre, année, nombre de pages), de demander une suggestion (livre le plus récent qui contient le moins de pages) et de quitter le programme. Implémentez ça au moyen d'une file de priorité.

Bonus : trouvez un moyen pour que les livres qui sont dans la bibliothèque depuis un certain temps acquièrent de la priorité au fur et à mesure du temps. Peut-être en ajoutant une valeur de priorité à chaque livre basée sur l'année et le nombre de pages, et qui est modifiée à chaque fois qu'un nouveau livre est ajouté pour le rendre de plus en plus prioritaire.