1. Simulateur de navigation (Back / Forward)

Ce projet simule le fonctionnement de la navigation dans un navigateur web : une pour gérer les pages visitées précédemment et une autre pour les pages à venir. Lorsqu'un utilisateur visite une nouvelle page, l'URL est ajoutée à la pile des pages visitées, et la pile « avant » est vidée, car une nouvelle branche de navigation est entamée.

Quand l'utilisateur clique sur « Retour », la page actuelle est déplacée dans la pile « avant », et la page précédente est dépilée de la pile « retour » pour devenir la page courante. Le bouton « Suivant » fonctionne de manière inverse : il déplace la page du haut de la pile « avant » dans la pile « retour ». Ce projet permet de bien comprendre l'utilisation des piles pour gérer l'historique non linéaire des actions utilisateur.

2. Système de réservation de billets

Ce projet propose de simuler un système de réservation de billets pour un événement, un moyen de transport ou un spectacle. Chaque réservation est représentée par un nœud de liste chaînée contenant des informations telles que le nom du client, la date, le siège ou le numéro de réservation. Les utilisateurs peuvent ajouter de nouvelles réservations, les annuler ou les consulter dynamiquement.

Ce système met l'accent sur la gestion efficace de la mémoire et la flexibilité des structures dynamiques. La liste chaînée permet d'insérer ou de supprimer facilement une réservation sans avoir à déplacer d'autres données, contrairement à un tableau. Le projet est idéal pour apprendre les opérations de base sur les listes chaînées comme l'insertion en tête, en queue ou à une position donnée.

3. Vérificateur de parenthèses équilibrées

Ce projet consiste à analyser une expression mathématique ou logique afin de vérifier si les parenthèses sont correctement ouvertes et fermées. Lorsqu'une parenthèse ouvrante est rencontrée, elle est empilée. Lorsqu'une parenthèse fermante apparaît, le programme vérifie si une parenthèse ouvrante correspondante peut être dépilée. Si à la fin du traitement la pile est vide, alors les parenthèses sont équilibrées.

Ce mécanisme est un exemple classique de la puissance des piles dans l'analyse syntaxique. Il peut être étendu pour vérifier d'autres types de symboles (crochets, accolades) et constitue une base idéale pour des applications comme les compilateurs ou les éditeurs de code.

4. Gestion de patients à l'hôpital (file d'attente)

Ce projet simule le fonctionnement d'une salle d'attente dans un hôpital où chaque patient est pris en charge selon l'ordre d'arrivée. Lorsqu'un patient arrive, il est ajouté en fin de file. Quand un médecin est disponible, il appelle le patient en tête de la file. Le programme gère ainsi les arrivées, les départs et peut afficher les patients en attente.

Ce système de file reflète fidèlement la gestion du temps réel et de la priorité naturelle. Il peut être enrichi en ajoutant des files par niveau d'urgence, ou un système de tickets pour une meilleure visualisation de la file.

5. Simulateur d'imprimante (queue de tâches)

Ce projet modélise le fonctionnement d'une imprimante partagée dans un réseau. Chaque document envoyé à l'impression est ajouté à une file de tâches. L'imprimante traite ensuite les documents dans l'ordre d'arrivée. L'utilisateur peut consulter l'état de la file, ajouter une tâche ou en supprimer une.

Ce simulateur met en avant la nature séquentielle du traitement des tâches et l'importance de la gestion de la file d'attente dans un environnement multi-utilisateur. Il peut être amélioré pour permettre la priorisation des documents selon leur taille, leur type (texte, image), ou selon l'utilisateur.

6. Historique d'actions dans un éditeur de texte (Undo/Redo)

Ce projet permet d'implémenter les fonctions "Annuler" et "Rétablir" dans un éditeur de texte. Chaque fois qu'un utilisateur effectue une action (ajouter un caractère, supprimer, coller), celle-ci est stockée dans une pile d'actions. Lorsqu'il appuie sur "Undo", l'action est dépilée de la pile des actions et poussée dans une autre pile "Redo".

Le système permet de revenir en arrière dans l'historique d'édition, puis de réappliquer les actions annulées si besoin. Ce projet est fondamental pour comprendre comment conserver et restaurer l'état d'une application.

7. Trie de nombres avec une file

Ce projet consiste à trier une liste de nombres à l'aide uniquement de structures de type file. Un algorithme possible est l'utilisation de plusieurs files pour trier les nombres par "paquets" ou "buckets" selon leur chiffre des unités, dizaines, etc. (tri radix par exemple).

Ce type de projet oblige l'étudiant à manipuler des files de manière non conventionnelle et à trouver des stratégies innovantes pour contourner l'absence d'accès direct ou de tri classique. Cela permet également d'approfondir la logique algorithmique derrière le tri distribué.

8. Liste chaînée pour gérer un carnet de contacts

Ce projet met en place un carnet de contacts dynamique où chaque contact est représenté par un nœud contenant un nom, un numéro de téléphone et une adresse e-mail. Les opérations disponibles incluent l'ajout, la suppression, la recherche et l'affichage de tous les contacts.

Grâce à la structure en liste chaînée, les contacts peuvent être stockés dynamiquement sans gaspillage de mémoire, et manipulés facilement. Ce projet illustre l'utilisation pratique des listes chaînées pour modéliser des bases de données légères et flexibles.

9. Simulation d'une file d'attente dans un parc d'attractions

Ce projet simule le comportement des visiteurs qui attendent leur tour pour accéder à une attraction. Les visiteurs arrivent à différents intervalles, et l'attraction peut accueillir un certain nombre de personnes à la fois. Le système gère dynamiquement les entrées et sorties en respectant la file.

Des événements aléatoires comme des départs anticipés, ou des fermetures temporaires de l'attraction peuvent être intégrés pour rendre la simulation plus réaliste. C'est un excellent projet pour visualiser les systèmes de files dans les environnements réels.

10. Jeu de cartes (pile de cartes)

Ce projet modélise un jeu de cartes où les cartes sont empilées et manipulées à l'aide d'une pile. Chaque carte peut être tirée du haut de la pile (comme lorsqu'on pioche dans un paquet), puis jouée ou remise. L'utilisateur peut mélanger, distribuer, ou tirer des cartes aléatoirement.

Ce projet permet d'appliquer la structure de pile à un contexte ludique tout en explorant la logique de distribution, de rotation et de gestion de paquets. Il peut être adapté pour simuler des jeux réels (UNO, bataille, solitaire, etc.).

11. Simulateur de service client en ligne

Ce projet simule un centre de support client où les tickets des utilisateurs sont traités dans l'ordre d'arrivée. Chaque ticket est ajouté à une file d'attente, et les agents de support les traitent un par un. Cela permet de modéliser le comportement d'un système de gestion de tickets en temps réel.

L'objectif est de comprendre le fonctionnement des files et de leur application dans des systèmes de traitement séquentiel. Ce projet peut être enrichi en ajoutant des priorités aux tickets ou en simulant plusieurs agents de support.

12. Gestionnaire de commandes dans un restaurant

Dans ce projet, chaque commande passée par un client est ajoutée à une file d'attente pour être préparée et servie. Les commandes sont traitées dans l'ordre d'arrivée, reflétant le fonctionnement réel d'une cuisine de restaurant.

Ce projet illustre l'utilisation des files pour gérer des tâches séquentielles et peut être étendu pour inclure des priorités, comme les commandes à emporter ou les commandes urgentes.

13. Système de réservation de billets

Ce projet utilise une liste chaînée pour gérer les réservations de billets. Les utilisateurs peuvent ajouter, annuler ou consulter des réservations. Chaque nœud de la liste représente une réservation avec des détails tels que le nom du client, la date et le numéro de siège.

L'objectif est de comprendre la gestion dynamique de données et la manipulation de listes chaînées. Ce projet peut être adapté pour différents types de réservations, comme des concerts ou des voyages.

14. Convertisseur de nombres décimaux en binaire

Ce projet consiste à convertir un nombre entier décimal en sa représentation binaire en utilisant une pile. Le nombre est divisé par 2 à chaque étape, et le reste est empilé. Ensuite, les éléments de la pile sont dépilés pour obtenir le nombre binaire.

Ce projet aide à comprendre le fonctionnement des piles et leur utilisation dans les algorithmes de conversion. Il renforce également la compréhension des systèmes de numération.

15. Jeu du Palindrome

Ce projet vérifie si un mot ou une phrase est un palindrome en utilisant une pile pour inverser l'ordre des lettres. Le mot est empilé caractère par caractère, puis dépilé pour obtenir sa version inversée. Si le mot original et le mot inversé sont identiques, c'est un palindrome.

Ce projet est utile pour comprendre la manipulation des piles et la comparaison de chaînes de caractères. Il peut être étendu pour ignorer les espaces et la casse des lettres.

16. Gestionnaire de favoris Internet

Ce projet utilise une liste chaînée pour gérer les favoris Internet. Les utilisateurs peuvent ajouter, modifier, supprimer et parcourir des liens favoris. Chaque nœud de la liste contient des informations telles que le nom du site et l'URL.

Ce projet permet de pratiquer la gestion dynamique de données et la manipulation de listes chaînées. Il peut être amélioré en ajoutant des fonctionnalités de recherche ou de classement des favoris.

17. Mini-calculatrice utilisant une pile

Ce projet implémente une mini-calculatrice qui utilise une pile pour évaluer des expressions en notation postfixée (notation polonaise inversée). Les opérandes sont empilés, et les opérateurs effectuent des opérations sur les éléments de la pile.

Ce projet aide à comprendre l'évaluation des expressions et l'utilisation des piles dans les calculs. Il peut être étendu pour inclure plus d'opérateurs et gérer les erreurs d'expression.

18. Simulation d'une machine de tri postal

Ce projet simule une machine de tri postal qui trie automatiquement les lettres en fonction de leur destination. Chaque lettre est ajoutée à une file correspondant à sa destination. Les lettres sont ensuite traitées dans l'ordre d'arrivée pour chaque destination.

Ce projet illustre l'utilisation des files pour gérer des tâches de tri et de distribution. Il peut être adapté pour simuler d'autres systèmes de tri, comme les centres de distribution de colis.

19. Système de notification par ordre de priorité

Ce projet gère des notifications en les plaçant dans des files différentes selon leur niveau de priorité (haute, normale, basse). Les notifications sont traitées en donnant la priorité aux files de niveau supérieur.

Ce projet permet de comprendre la gestion des priorités et l'utilisation de multiples files. Il peut être étendu pour inclure des délais d'expiration des notifications ou des filtres personnalisés.

20. Playlist musicale

Ce projet utilise une liste chaînée pour gérer une playlist musicale. Les utilisateurs peuvent ajouter, supprimer et parcourir des chansons dans la playlist. Chaque nœud de la liste contient des informations sur une chanson, comme le titre et l'artiste.

Ce projet aide à comprendre la manipulation de listes chaînées et la gestion dynamique de données. Il peut être amélioré en ajoutant des fonctionnalités de recherche ou de lecture aléatoire.

TRAVAIL A FAIRE :

- **Fournir un cahier de charges (la structure vous est indiquée en annexe du présent document)**
- **Produire un code qui marche avec interface graphique sans base de données. La bibliothèque Qt vous permettra de réaliser votre (vos) intreface graphique.**
- **Préparer une présentation PowerPoint**

ANNEXE

Structure Recommandée d'un Cahier des Charges (SRS - Software Requirements Specification) :

1. Introduction

1.1. But du Document : Décrit l'objectif de ce cahier des charges, le public visé (qui lira et utilisera ce document), et l'utilisation prévue du document (par exemple, base pour la conception, le développement, les tests, projet d'étude, etc.).

1.2. Portée du Projet : Délimite clairement le périmètre du projet, ce qui sera inclus et ce qui ne le sera pas. Fournit un aperçu général du système ou du produit à développer.

1.3. Définitions, Acronymes et Abréviations : Liste et définit tous les termes techniques, acronymes et abréviations utilisés dans le document pour assurer une compréhension uniforme.

1.4. Références : Liste tous les documents externes référencés dans le cahier des charges (par exemple, documents de spécification du client, normes industrielles, contrats).

1.5. Vue d'Ensemble du Reste du Document : Fournit un aperçu de l'organisation du reste du document pour faciliter la lecture.

2. Description Générale

2.1. Contexte du Produit : Décrit le contexte dans lequel le produit sera utilisé, son environnement d'exploitation, et sa relation avec d'autres systèmes existants.

2.2. Besoins des Utilisateurs et Objectifs du Projet : Présente les besoins des utilisateurs finaux et les objectifs commerciaux ou stratégiques que le projet vise à atteindre. Cela peut inclure des scénarios d'utilisation de haut niveau.

2.3. Caractéristiques du Produit : Résume les principales fonctionnalités et capacités du produit à développer.

2.4. Contraintes : Identifie les contraintes qui affectent le développement du produit, telles que les contraintes budgétaires, temporelles, technologiques, réglementaires, de sécurité, de performance, etc.

- **2.5. Hypothèses et Dépendances :** Liste les hypothèses faites lors de la rédaction du cahier des charges et les dépendances externes qui pourraient impacter le projet.

3. Exigences Spécifiques

3.1. Exigences Fonctionnelles (ERF) : Décrivent ce que le système doit faire. Elles sont généralement organisées par fonctionnalité, cas d'utilisation ou processus métier. Pour chaque exigence fonctionnelle, il est recommandé d'inclure :

- **Identifiant unique :** Pour la traçabilité.
- **Description détaillée :** Claire et non ambiguë du comportement attendu.
- **Acteurs impliqués (le cas échéant).**
- **Préconditions :** Ce qui doit être vrai avant que la fonctionnalité puisse être exécutée.
- **Déclencheur :** L'événement qui initie la fonctionnalité.
- **Séquence normale des événements :** Les étapes du processus dans des conditions normales.
- **Flux alternatifs ou exceptions :** Ce qui se passe en cas d'erreurs ou de conditions inhabituelles.
- **Postconditions :** Ce qui est vrai une fois que la fonctionnalité a été exécutée.
- **Priorité :** Importance de l'exigence (par exemple, essentielle, importante, souhaitable).
- **Critères d'acceptation :** Conditions spécifiques qui doivent être satisfaites pour considérer que l'exigence est implémentée correctement.

3.2. Exigences Non Fonctionnelles (ERNF) : Décrivent les qualités du système, comment il doit fonctionner. Elles sont souvent catégorisées comme suit :

- **Exigences de Performance :** Temps de réponse, débit, capacité, utilisation des ressources.
- **Exigences de Sécurité :** Confidentialité, intégrité, disponibilité, authentification, autorisation.
- **Exigences de Fiabilité :** Disponibilité, tolérance aux pannes, récupérabilité.
- **Exigences d'Ergonomie et d'Interface Utilisateur :** Facilité d'utilisation, accessibilité, esthétique.
- **Exigences de Maintenabilité :** Facilité de modification, de correction, d'adaptation.

- **Exigences de Portabilité :** Capacité à fonctionner sur différentes plateformes ou environnements.
- **Exigences de Conformité :** Respect des normes légales, réglementaires ou industrielles.

3.3. Exigences d'Interface Externe : Décrivent les interactions du système avec des entités externes :

- **Interfaces Utilisateur (UI) :** Spécifications de l'interface graphique, des interactions utilisateur.
- **Interfaces Matérielles :** Interactions avec des dispositifs physiques.
- **Interfaces Logicielles :** Interactions avec d'autres systèmes ou applications (APIs, protocoles).
- **Interfaces de Communication :** Protocoles et formats de données utilisés pour la communication.

3.4. Autres Exigences (si nécessaire) : Peut inclure des exigences spécifiques au projet qui ne rentrent pas dans les catégories précédentes.

4. Préparation à la Livraison

4.1. Plan de Déploiement : Description de la manière dont le système sera déployé dans son environnement de production.

4.2. Exigences de Formation : Besoins en formation pour les utilisateurs et les administrateurs du système.

4.3. Exigences de Documentation : Types de documentation à fournir (manuel utilisateur, guide d'installation, documentation technique, etc.).

4.4. Exigences de Maintenance : Spécifications relatives à la maintenance future du système.

5. Annexes (Optionnel)

- **Glossaire détaillé.**
- **Prototypes ou maquettes d'interface.**
- **Informations supplémentaires pertinentes.**

Qualités d'un Bon Cahier des Charges (selon IEEE 830 et ISO/IEC/IEEE 29148) :

- **Complet :** Toutes les exigences significatives sont incluses.
- **Non ambigu :** Chaque exigence a une seule interprétation possible.
- **Vérifiable :** Il doit être possible de déterminer si le système satisfait à l'exigence.
- **Cohérent :** Aucune exigence n'est en conflit avec une autre.
- **Modifiable :** Il doit être possible de modifier le document tout en conservant sa cohérence et sa traçabilité.
- **Traçable :** Chaque exigence doit pouvoir être reliée à son origine (besoin utilisateur) et à sa réalisation (conception, code, tests).
- **Priorisé :** Les exigences sont classées par ordre d'importance.
- **Clair et concis :** Facile à lire et à comprendre.