IUT des Pays de l'Adour - RT1
M1109 - Projet informatique R&T

Algorithmique / Python
Centrale domotique

Manuel Munier

Version 4 décembre 2020 (10:00)

Expression des besoins

On désire gérer par des moyens informatiques (par exemple une programme écrit en Python...) les données envoyées par les différents capteurs d'une installation domotique. Les capteurs enregistrent leurs données au niveau de la centrale. Cette dernière propose ensuite différents services : historique complet, données collectées sur une période donnée, valeur minimum/maximum/moyenne d'un capteur (éventuellement sur une période donnée), pour tous les capteurs dernière valeur collectée à un instant t, interpolation de la valeur d'un capteur à un instant t, etc.

L'objectif de ce projet informatique à la fin du S1 est d'asseoir les compétences de base en algorithmique acquises au cours du module M1207 "bases de la programmation". Nous avons pour cela décidé d'ajouter des créneaux horaires pour un volume de 9h encadrées + 12h en autonomie. Vous travaillerez individuellement afin que chacun d'entre vous puisse prendre conscience de ses (éventuelles) lacunes, rattraper son (éventuel) retard et progresser en algorithmique sans devoir attendre son binôme/groupe/promo.

Ce projet doit être et sera noté. Pour cela, les enseignants vous évalueront au fur et à mesure tant du point de vue du travail fourni que de votre attitude, ceci afin de savoir "à quoi s'attendre" pour les modules d'informatique du S2 ☺. Nous vous conseillons donc vivement de prendre ce projet au sérieux d'autant que 1) ces notions d'algorithmique vous seront essentielles pour la suite de votre cursus (et pas qu'en DUT !) et 2) c'est le département R&T qui fait l'effort de financer ces heures sur ses fonds propres. D'après le PPN du DUT R&T les notions "de base" en algorithmique susnommées sont :

- variables, structures de données (tableaux, tuples, listes)
- structures de contrôle (tests, boucles, structures imbriquées,...)
- sous-programmes (fonctions, avec passage de paramètres et retour de résultat, visibilité des variables)
- concevoir un algo à partir d'un énoncé, le traduire en Python
- ~~compiler~~, corriger et tester un programme

Consignes de programmation

Voici quelques consignes (de bon sens) pour prendre de bonnes habitudes en programmation :

- Pensez à commenter votre code.
- Au début de chaque fonction indiquez en commentaire le rôle de chaque variable passée en paramètre et le type du résultat qui sera retourné.
- Une fonction de doit pas effectuer d'entrées/sorties, c'est-à-dire ne doit pas demander à l'utilisateur de saisir des valeurs au clavier ou lui afficher des informations dans le terminal. Sauf, bien évidemment, les fonctions dont le rôle est explicitement de saisir une valeur ou d'afficher une information...
- Prenez l'habitude que vos fonctions s'exécutent jusqu'à la fin de leur code dont la dernière instruction est le `return`. Évitez à tout prix de faire un `return` dans une boucle ou dans un test ; ça fonctionne, certes, mais c'est source de gros problèmes lors du debugage...

Travail à réaliser

D'un point de vue technique, ce projet sera constitué d'un seul programme (composé bien évidemment des diverses fonctions) exécuté en ligne de commande dans un terminal texte. Dans un premier temps, il n'y aura aucune interaction avec l'utilisateur : toutes les "actions" seront directement hard-codées dans le programme principal, ce qui d'ailleurs vous facilitera la phase de test. Dans un second temps uniquement, les interactions avec l'utilisateur se feront au travers d'un menu textuel (nous vous guiderons pour cela...) où le choix d'une fonctionnalité par l'utilisateur déclenchera l'exécution d'une fonction dédiée.

Les informations à stocker seront donc :

- la **date** de collecte sous la forme de 3 entiers jour/mois/année
- l'**heure** de collecte (3 entiers heure/minutes/secondes)
- l'**identification du capteur** ; il s'agit d'une chaîne de caractères (NB : attention donc aux erreurs de saisies : majuscules/minuscules, espaces,...)
- la **valeur** enregistrée (un numérique, ou une chaîne de caractères si vous voulez quelque chose de plus général)
- le **type** de la donnée (une chaîne de caractères pour indiquer s'il s'agit d'une température, d'une luminosité, d'un taux d'humidité, etc.)

Toujours par soucis de simplification, la "base de données" pour le stockage sera représentée sous la forme d'un (grand) tableau en mémoire. Nous supposerons que, par construction, toutes les colonnes de votre tableau seront remplies "correctement". Par exemple, lorsque l'on ajoutera une nouvelle valeur, i.e. une nouvelle ligne dans le tableau, les triplets pour la date et l'heure de collecte représenteront bien des dates et des heures valides. Peu importe l'ordre des lignes, le tri/classement chronologique sera effectué par vos fonctions après avoir appliqué les filtres adéquats.

Voici dans les grandes lignes votre planning de travail. Bien évidemment, à chaque étape il vous faudra écrire un petit programme de test afin de valider les fonctions que vous aurez écrites.

1. Implémenter la fonctionnalité permettant d'ajouter une nouvelle valeur (envoyée par un capteur). Cette fonction prend en paramètres les différentes informations, construit la nouvelle ligne et l'ajoute au tableau.

2. Implémenter la fonctionnalité permettant d'afficher, dans l'ordre d'insertion, toutes les données collectées du tableau passé en paramètre. Pour simplifier, point de présentation graphique ; l'affichage se fera de manière textuelle dans la console.

3. Implémenter la fonctionnalité permettant de filtrer un tableau de stockage pour ne conserver que les lignes concernant un capteur donné. Cette fonction prendra en entrée (aka en paramètre) un tableau de stockage ainsi que l'identifiant du capteur, et retournera comme résultat un nouveau tableau de stockage (même "format") contenant uniquement les lignes retenues.

4. Idem pour un intervalle de temps donné (entre telle date telle heure et telle date telle heure), pour un type de capteur donné, etc... Le fait que ces fonctions de sélection fonctionnent à la manière de filtres, c'est-à-dire qu'elles prennent en entrée un tableau de stockage et un critère pour retourner en sortie un tableau de stockage, va nous permettre "d'emboîter" ces filtres pour réaliser des recherches multi-critères.

5. Écrire une fonction permettant de tirer par ordre chronologique le tableau de stockage passé en paramètre. L'objectif de ce projet étant de vous perfectionner en algorithmique, il vous est bien évidemment **interdit d'utiliser la fonction `sort`** disponible en Python sur les tableaux ; à vous d'implémenter votre propre algorithme de tri ! (cf. tri par sélection, tri à bulles, etc.)

6. Écrire le squelette de votre menu principal : affichage des actions disponibles, saisie du choix de l'utilisateur en vérifiant la validité de ce choix, appel à la fonction adéquate, puis retour au menu. La fonction adéquate dont nous parlons ici devra faire 3 choses :
   a) lecture des paramètres nécessaires
   b) appel de la fonction de traitement (une des fonctions programmées dans les points précédents)
   c) affichage du résultat

Fonctionnalités supplémentaires

Au cas où... :

- Améliorer l'affichage d'un tableau de stockage en "découpant" jour par jour. Par exemple ajout d'une ligne de titre au début de la journée, et saut d'une ligne à la fin.
- Implémentation de différentes fonctions statistiques (liste non exhaustive) :
  a) valeur minimum/maximum/moyenne d'un capteur sur la totalité des données collectées (il y a donc 3 fonctions à écrire)
  b) valeur minimum/maximum/moyenne d'un capteur sur un intervalle de temps donné (entre telle date telle heure et telle date telle heure)
  c) pour chaque jour, nombre de valeurs collectées pour chaque capteur
  d) à un instant t, dernière valeur collectée pour un capteur donné
  e) à un instant t, interpolation (linéaire) de la valeur d'un capteur en fonction des 2 valeurs (avant t et après t) les plus proches
  f) ...
- Et pourquoi pas... utilisation d'une librairie graphique permettant de générer de graphes à partir de jeux de données. Une solution serait la librairie Matplotlib et notamment l'exemple *"Grouped bar chart with labels"*. Sur l'exemple *"Simple Plot"* il est également indiqué que l'on peut enregistrer le graphe au format PNG avec un `fig.savefig("test.png")`