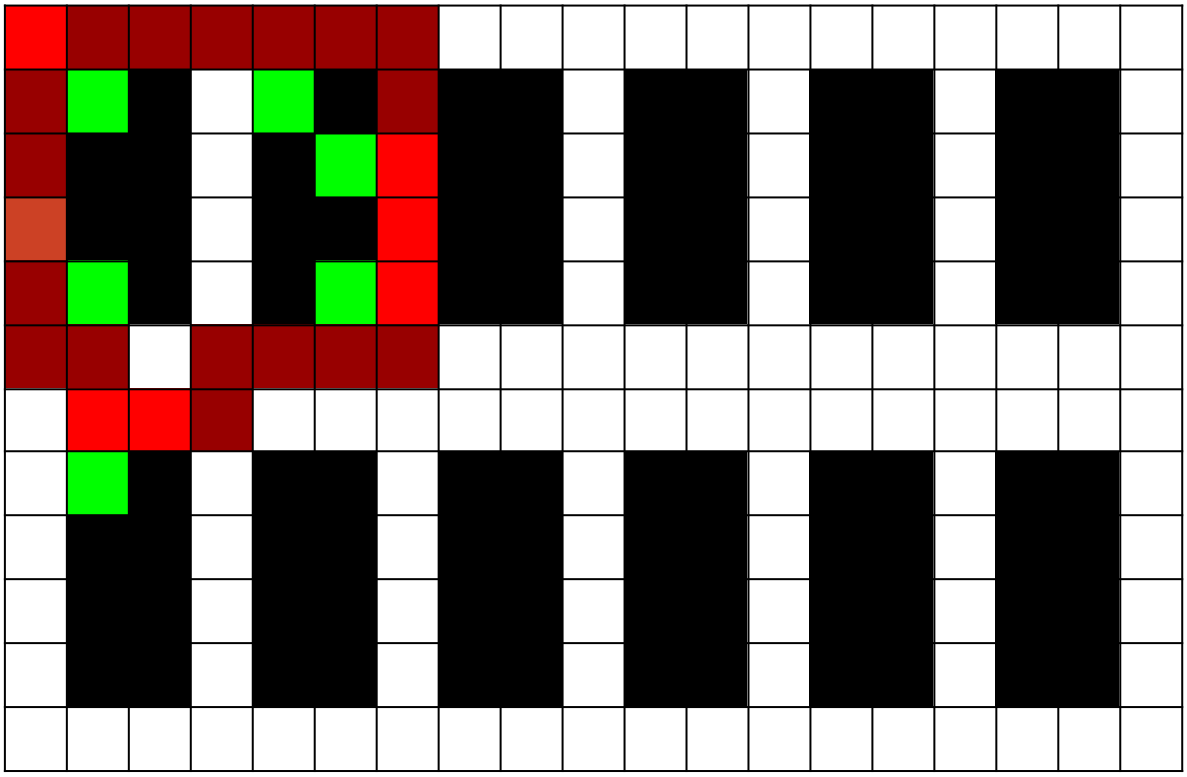
Rapport de soutenance 1

# Perfect Course

Votre allié pour optimiser votre temps et vos forces



https://perfectcourse.neocities.org/

Erwan Maigne-Montamat
Jean-Loup de Beauminy
Mathias Lecoeur
Thomas Tayrac

Sommaire:

# 1. Les adaptations du planning:

Au vu des circonstances spécifiques sur Toulouse (confirmation quelque peu tardive) nous avons également pris un retard de deux semaines.
Il faudra donc considérer que l'ancien planning reste d'actualité mais avec un décalage de deux semaines (un récapitulatif mis à jour sera disponible en annexes).

# 2. Présentation du groupe

## 2.1) Jean Loup de Beauminy

Je me suis intéressé à l'informatique en seconde pendant les cours de maths où on programmait sur nos calculatrices. Après cela, j'ai passé mon temps à apprendre d'autres langages comme css/html, javascript, python ou encore le c. Je pars donc avec un petit avantage sachant que je connaissais déjà le c et que le projet est dans ce langage. Cependant, le projet implique des connaissances en interfaces graphiques et en création de chemin . Ces parties restent un défi que j'ai hâte de réaliser.

## 2.2) Mathias Lecoeur:

Étant la seconde fois que je fais la spé de l'Epita j'ai de solides bases en C, de plus ce projet sera à mes yeux, l'opportunité d'obtenir de nouvelles compétences, de combler mes lacunes mais aussi de mettre à profit les connaissances acquises en prépa classique.

## 2.3) Erwan Maigne-Montamat:

Je suis intéressé par l'informatique depuis mes 10ans et à l'époque ou je jouais beaucoup à minecraft, puis a suivi le développement de plug in et de mod avant de partir sur un bac S science de l'ingénieur avec une spécialité ISN.
Enfin je suis très intéressé par tout ce qui est du domaine de l'algorithmique, le pathfinding faisant partie de ce projet me motive fortement.

## 2.4) Thomas Tayrac:

Personnellement, je suis intéressé par la programmation et l'algorithmie depuis la première lorsque j'ai découvert le langage python en cours d'ICN. C'est une tournure d'esprit un peu spéciale et complexe mais cela ouvre les portes a tellement de possibilités, de solutions à des problèmes etc. On se sert de ce projet justement pour résoudre un problème de perte inutile d'énergie et de temps d'un employé ; selon moi c'est quelque chose d'utile et si on parvient à réaliser ce projet dans les moindres détails, j'en serais fier. Voilà pourquoi j'éprouve une motivation particulière à l'idée de faire ce projet comme il se doit.

# 3. Le compte rendu d’avancement

## 3.1. La structure graph:

Nous avons donc implémenté une structure graph globalement similaire à celle vue en algorithmie, néanmoins nous lui avons ajouté des informations complémentaires relatives aux produits stockés, aux coordonnées des nœuds dans l’entrepôt. Il nous a également fallu créer des fonctions classiques pour pouvoir modifier le graph, la subtilité étant qu’au vu de l’absence de classe en C il nous a fallu récupérer les données du graph directement pour les modifier par références.
Enfin puisque cette structure graph est la schématisation d’un entrepôt nous avons décidé de lui rajouter d'autres données tel que l'index de tous les produits en stock ou leur poids.

## 3.2. Chargement du graph à partir d’un fichier texte:

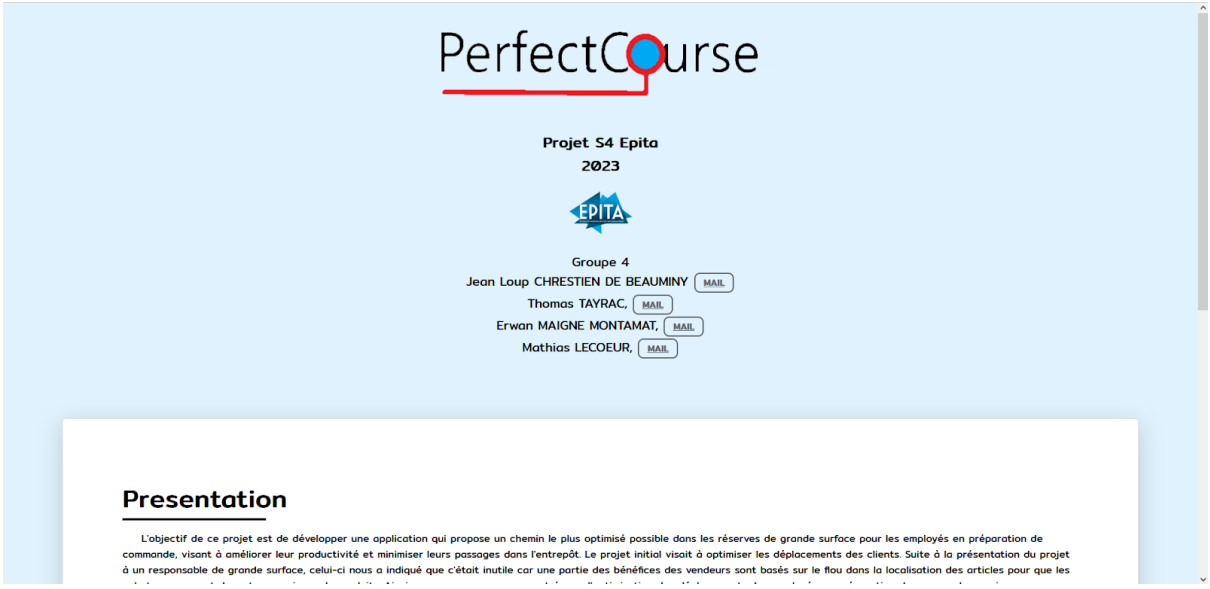
C’est une étape dont nous avions négligé l’importance dans la création du cahier des charges mais il est pourtant évident que sauvegarder un entrepôt et le récupérer est un besoin fondamental, non seulement ceci, mais nous en avons également sous-estimé la difficulté. Néanmoins après de multiples embûches nous avons le chargement d’un fichier de sauvegarde, le parsing de ce dernier et la reconstruction d’un graph à partir dudit fichier texte, pour la partie sauvegarde nous avons actuellement un print dans la console qui n’est pas encore tout à fait conforme au format de fichier de sauvegarde que nous utilisons, néanmoins nous n’en sommes pas loin. Cette partie sera approfondie dans le compte rendu des contributions personnelles dans la section de Thomas.

## 3.3. Le site web

Pour ce qui est du site web, ce dernier est indiqué sur la première page de ce rapport, l'adresse est la suivante:

https://perfectcourse.neocities.org/

C'est pour le moment une ébauche du site final en HTML CSS uniquement. Le site est hébergé sur neocities (un hébergeur gratuit). Vous pourrez retrouver dessus un lien vers le github du projet et ce dit rapport au format numérique ainsi que tous les documents utiles au projet.




## 3.4. QRcode et Localisation

Nous avons commencé à travailler sur la localisation des employés dans l'entrepôt, le principe étant plutôt simple en théorie: Lorsque l'employé récupère un produit il scan sa référence, étant donné que nous avons archivé la position de chaque produit dans l'entrepôt nous pouvons en déduire la position de l'employé, néanmoins la difficulté est maintenant de reconnaître la référence aisément, rappelons que les OCRs ont été un projet entier au troisième semestre, nous n'excluons pas

d’ignorer la partie OCR. Le lien entre la référence et sa position est néanmoins déjà établi.

## 3.5. Interface graphique

L'objectif de cette implémentation graphique est de visualiser un graphe en deux dimensions représentant une configuration de rayons et d'allées dans un magasin. Le but est de permettre aux utilisateurs de mieux comprendre la disposition de ces rayons dans le magasin et de faciliter notre propre compréhension de la structure du graphe.

Lorsque le programme est exécuté, une fenêtre graphique s'ouvre, dans laquelle le graphe est affiché. La bibliothèque OpenGL est utilisée pour créer la fenêtre et effectuer le rendu graphique.

La position des noeuds dans la fenêtre graphique est déterminée par leurs coordonnées X et Y dans le graphe. Les coordonnées sont calculées en multipliant les coordonnées des intersections dans le graphe par des facteurs d'échelle, puis en les ajustant pour les placer dans la fenêtre graphique.

Enfin, pour faciliter la lecture du graphe, différentes couleurs et épaisseurs de traits sont utilisées pour représenter les différents types de connexions entre les intersections.
(p.18)

# 4. Répartition des tâches

## Répartition initiale des tâches:

| Tâche | Jean Loup | Thomas | Erwan | Mathias |
|---|---|---|---|---|
| Implémentation graph en C | X | X | X | X |
| Codage base du site | X | | | |
| Création outil de modélisation entrepôt | | X | X | X |
| Localisation grâce au QR code | X | | X | |
| Reconnaissance QR code | | X | | X |
| Interface de démonstration | X | X | X | X |
| Préparation première soutenance | X | X | X | X |
| Algorithme plus court chemin | X | | X | |
| Interface d'ajout de référence | | X | | X |
| Création d'un entrepôt type | | X | | |
| Algorithme de recherche de chemin selon le poids | X | | X | |
| Création design définitif site | X | | X | |
| Codage du design définitif | X | | | |

| | | | | |
|---|---|---|---|---|
| du site | | | | |
| Débogage avant la seconde soutenance | | X | | X |
| Localisation “en temps réel” grâce à une simulation de triangulation bluetooth | X | X | X | X |
| Détection égarement employé | X | X | | |
| Préparation seconde soutenance | X | X | X | X |
| Correcteur d’itinéraire | | | X | X |
| Création du parser pour implémenter la liste de produit à aller chercher | | X | | X |
| Générateur aléatoire de commandes | X | | X | |
| Tests de la version “finie” | X | X | X | X |
| Amélioration du design final | X | X | X | X |
| Débogage avant la dernière soutenance- | X | X | X | X |
| Préparation dernière soutenance | X | X | X | X |

# Compte rendu individuel d’avancement

## Mathias Lecoeur

Mon rôle a surtout été d’aider sur un petit tout, allant de la rédaction des documents type cahier des charges rapports de soutenance à des propositions et des avis sur les diverses implémentations en termes de code.

## Thomas Tayrac

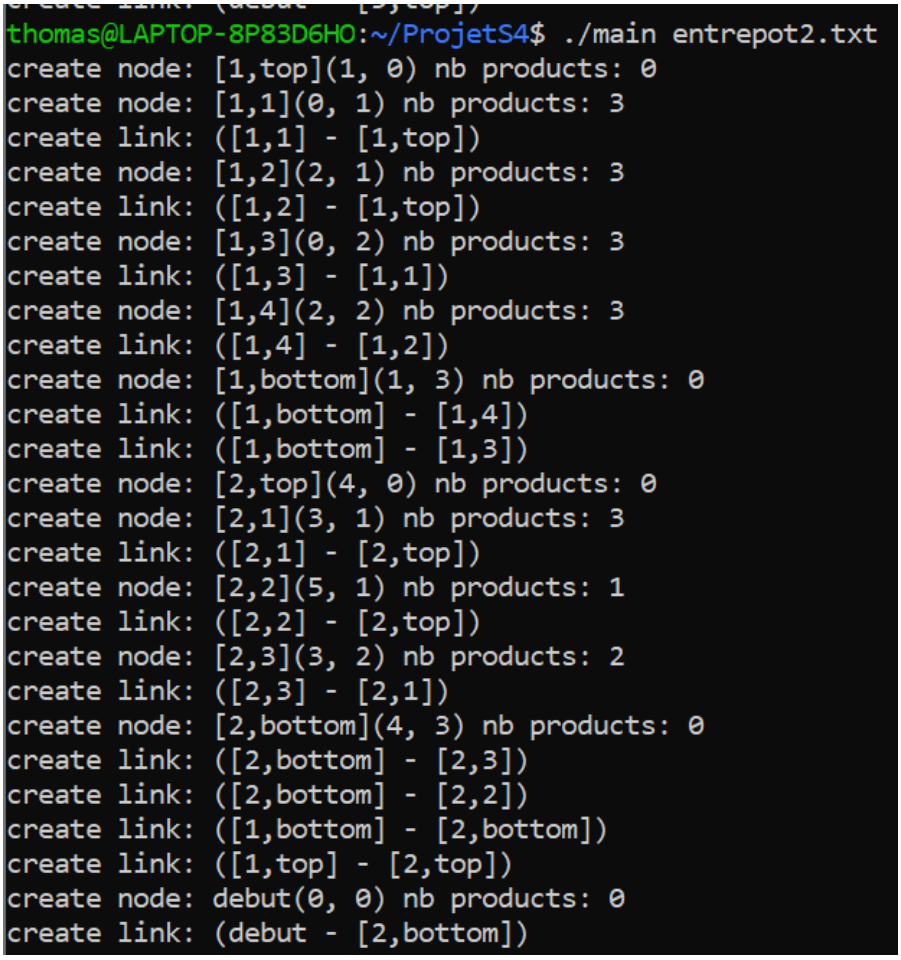
pour ma part je me suis chargé de la partie lecture de fichier, ou plus précisément du passage du fichier texte au graphe sur lequel on va pouvoir travailler par la suite. On aura besoin donc en entrée d’un fichier texte (représentant l'entrepôt) fourni par le client. Celui-ci devra fournir à la fois chaque rayon avec ses différentes étagères dans lesquelles se situent les articles (3 précisément, c’est le nombre maximum que nous avons choisi pour le moment) et les liens entre rayon (les chemins que l’on peut prendre pour aller d’un rayon à l'autre). Voici comment ce fichier devra être disposé :

```
thomas@LAPTOP-8P83D6HO:~/ProjetS4$ cat entrepot2.txt
aisle 1 top(1,0) bottom(1,3)
01(0,1) 12 34 03
02(2,1) 33 11 01
03(0,2) 13 54 22
04(2,2) 25 09 12

aisle 2 top(4,0) bottom(4,3)
01(3,1) 12 34 03
02(5,1) 10 -1 -1
03(3,2) 13 23 -1

aisle links:
1_bottom 2_bottom
1_top 2_top
debut(0, 0) 2_bottom
```

un rayon sera représenté par une ligne de ce style :
“aisle n top(x,y) bottom(x’,y’)”
-> top étant le haut du rayon (ou l’”entrée”) et bottom le bas (ou la “sortie”). En dessous de chaque rayon se situeront les lignes correspondant aux étagères. chaque ligne étagère contiendra : le numéro de l’étagère dans le rayon de 1 à n, les coordonnées (x,y) ainsi que les 3 références des produits présents dans l’étagère (-1 si étagère vide).

A la fin du fichier devra être présente une dernière partie : les liens entre les rayons. Cette partie commence à partir de la chaîne “aisle links:”. On peut comprendre sur l’exemple que le bottom du rayon 1 est atteignable du botom du rayon 2 (et inversement). Enfin une derniere position nous doit être connue, le début du parcours de l’employé que l’on doit relier au top/bottom le plus proche.

J’ai donc codé un programme qui, à partir d’un fichier de ce type renvoie le graphe correspondant. Ne pouvant pas afficher un graphe en console, je me suis dit qu’il était préférable d’afficher chaque étape de la construction du graphe. Voici donc le résultat lorsque que l’on applique mon code au fichier entrepot2.txt affiché précédemment :




Nous pouvons voir que 2 opérations différentes sont faites: création d'un nœud et création d'un arc. exemple : première ligne, on voit que le top du rayon 1 est créé aux coordonnées (1,0) avec 0 produits (logiquement, ce n'est pas une étagère).

Autre exemple : ligne numéro 3, on crée un lien entre l'étagère 1 du rayon 1 et le nœud top du rayon 1 (crée ligne 1). Et à la fin, on relie bien le nœud début au bottom du rayon 2 comme précisé dans le fichier texte.

Comment je suis parvenu à faire ça ?

étape 1 : coder la classe graph qui se présente comme ceci:

```
struct Graph
{
        int order; /* nb de noeuds */
        struct Node *tableNodes[512];
};

struct Graph G;

struct Node
{
        int boolShelf;/*si 0 -> intermediate, si 1 -> shelf, si 2 -> debut*/
        int aisle;
        int shelf;
        int boolTopBottom; /*si boolShelf = 0, btb = 1 si top, sinon 0*/
        struct Node *adjTab[10];
        int nbAdj;
        int X;
        int Y;
        int productsTab[3];
        int nbProducts;
};
```

On crée l'élément Node, dans le but de pouvoir les assembler pour former un graphe.
Ensuite, à partir :

- d'un "booléen" qui nous précise si le noeud est une étagère,
- du numéro de rayon,
- du numéro de l'étagère,
- d'un autre "booléen" qui nous précise si le noeud est un top ou un bottom (intéressant seulement si le noeud n'est pas une étagère)
- des coordonnées (X,Y) dans l'entrepôt
- des 3 références des articles (nous avons arbitrairement choisi qu'il y aurait entre 0 et 3 références par emplacement de stockage),

nous pouvons appliquer une fonction createNode qui, comme son nom l'indique, vient réellement créer le nœud associé (et l'ajouter au graphe).
Cette dernière se présente de la façon suivant :

```
struct Node *createNode(int boolShelf, int aisle, int shelf, int boolTopBottom, int X, int Y, int p1
, int p2, int p3)
{
        struct Node *node;
        node=malloc(sizeof(struct Node));

        node->boolShelf=boolShelf;
        node->aisle=aisle;
        node->shelf=shelf;
        node->boolTopBottom=boolTopBottom;
        node->nbAdj=0;

        node->X = X;
        node->Y = Y;

        node->productsTab[0]=p1;
        node->productsTab[1]=p2;
        node->productsTab[2]=p3;
        if (p1 != -1)
                node->nbProducts++;
        if (p2 != -1)
                node->nbProducts++;
        if (p3 != -1)
                node->nbProducts++;

        G.tableNodes[G.order]=node;
        G.order++;

        printf("create node: ");
        printNode(node);
        printf("(%d, %d) nb products: %d \n", node->X, node->Y, node->nbProducts);
        return node;
}
```

Pour pouvoir accéder au nœud du graphe, il a fallu que je crée 2 fonctions différentes qui retourne le nœud correspondant au rayon et à l'étagère. Pourquoi 2 fonctions ? pour séparer les nœuds étagère et les nœuds intermédiaires, car ils n'ont pas les mêmes attributs donc c'était un peu obligé. Voici ces 2 fonctions :

```
struct Node *getNodeShelf(int aisle, int shelf)
{
        int i=0;
        while (i<G.order)
        {
                if ((aisle==G.tableNodes[i]->aisle) && (shelf==G.tableNodes[i]->shelf))
                        return G.tableNodes[i];
                i++;
        }
        return NULL;
}

struct Node *getNodeIntermediate(int aisle, int boolTopBottom)
{
        int i=0;
        while (i<G.order) {
                if ((aisle==G.tableNodes[i]->aisle) && (boolTopBottom==G.tableNodes[i]->boolTopBotto
m))
                        return G.tableNodes[i];
                i++;
                }
        return NULL;
}
```

Après avoir traité les noeuds du graphe, il ne fallait pas oublier les arcs, voici donc une fonction qui crée ces arcs a partir de 2 objets noeuds ;

```
void createLink(struct Node *node1, struct Node *node2)
{
        if ((node1->nbAdj>=10)||(node2->nbAdj>=10))
                errx(1,"link creation error");
        node1->adjTab[node1->nbAdj]=node2;
        node1->nbAdj++;
        node2->adjTab[node2->nbAdj]=node1;
        node2->nbAdj++;

        printf("create link: (");
        printNode(node1);

        printf(" - ");
        printNode(node2);

        printf(")\n");
}
```

Et enfin, voici la fonction principale qui se sert donc des fonctions précédentes pour réaliser la lecture du fichier :

```c
struct Graph *FileRead(char *filename)
{
	FILE *file;
	char buffer[256];
	int aisle;int shelf;
	int article_1;int article_2;int article_3;
	int aisle_1;int aisle_2;
	char botop_1[50];char botop_2[50];
	struct Node *debut;
	int X1; int Y1; int X2; int Y2;


	file = fopen(filename, "r");
	if (file == NULL)
		errx(1,"opening error");

	while((fgets(buffer,256,file) != NULL) &&
			(sscanf(buffer,"aisle %d top(%d,%d) bottom(%d,%d", &aisle, &X1, &Y1, &X2, &Y2) == 5))
	{

		createNode(0,aisle,-1,1,X1,Y1,-1,-1,-1); /*noeud top*/
		while((fgets(buffer,256,file) != NULL ) &&
				(sscanf(buffer, "%d(%d,%d) %d %d %d", &shelf, &X1, &Y1,&article_1, &article_2, &article_3) == 6))
		{
			/*printf("shelf: %d a1: %d a2: %d a3: %d \n",shelf,article_1,article_2,article_3);*/
			createNode(1,aisle,shelf,-1,X1,Y1,article_1,article_2,article_3); /*noeud shelf*/
			if ((shelf == 1) || (shelf == 2))
				createLink(getNodeShelf(aisle,shelf),getNodeIntermediate(aisle,1));
			else
				createLink(getNodeShelf(aisle,shelf),getNodeShelf(aisle,shelf-2));
		}
		/*noeud bottom du dernier rayon*/
		createNode(0,aisle,-1,0,X2,Y2,-1,-1,-1);
		createLink(getNodeIntermediate(aisle,0),getNodeShelf(aisle,shelf));
		createLink(getNodeIntermediate(aisle,0),getNodeShelf(aisle,shelf-1));
	}
```

```c
	while((fgets(buffer,256,file) != NULL) && (sscanf(buffer,"%d_%s %d_%s", &aisle_1, botop_1, &aisle_2, botop_2) == 4))
	{
		if ((strcmp(botop_1,"top") == 0) && (strcmp(botop_1,"top") == 0))
			createLink(getNodeIntermediate(aisle_1,1),getNodeIntermediate(aisle_2,1));
		if ((strcmp(botop_1,"bottom") == 0) && (strcmp(botop_1,"top") == 0))
			createLink(getNodeIntermediate(aisle_1,0),getNodeIntermediate(aisle_2,1));
		if ((strcmp(botop_1,"top") == 0) && (strcmp(botop_1,"bottom") == 0))
			createLink(getNodeIntermediate(aisle_1,1),getNodeIntermediate(aisle_2,0));
		if ((strcmp(botop_1,"bottom") == 0) && (strcmp(botop_1,"bottom") == 0))
			createLink(getNodeIntermediate(aisle_1,0),getNodeIntermediate(aisle_2,0));
		/*printf("%d%s - %d%s \n", aisle_1, botop_1, aisle_2, botop_2);*/
		/* lien entre noeud top/bottom du rayon 1 et le noeud top/bottom du rayon 2*/
	}

	if (sscanf(buffer, "debut(%d,%d) %d_%s", &X1, &Y1, &aisle_1, botop_1) != 4)
		errx(1, "scanning error 5");
	/*printf("debut - %d_%s \n", aisle_1, botop_1);*/

	debut = createNode(2,-1,-1,-1, X1, Y1, -1, -1, -1);
	if (strcmp(botop_1,"top") == 0)
		createLink(debut,getNodeIntermediate(aisle_1,1));

	if (strcmp(botop_1,"bottom") == 0)
		createLink(debut,getNodeIntermediate(aisle_1,0));

	/* créer le noeud debut et le relier au noeud top/bottom du rayon*/
	fclose(file);
	return &G;
}
```

Il a fallu que je me serve principalement de la fonction sscanf qui va venir récupérer les informations d'une ligne (étant donné que je l'applique de la fonction fgets qui "sélectionne une ligne").

Je me sers de plusieurs boucles en coordination avec la structure du fichier texte pour interagir avec chacune des ligne de ce fichier et donc en ressortir un graphe.

# Erwan Maigne-Montamat

J'étais pour ma part chargé de la conception de la classe graph, néanmoins suite à des soucis de push et de manque de communication de ma part thomas n'a jamais eu la possibilité de la récupérer. Enfin du reste j'ai malgré tout participé à la réflexion autour de cette dernière et j'ai également réfléchi avec thomas à la meilleure manière d'ordonner l'enregistrement des graphs dans le fichier puisque initialement il semblait s'orienter sur quelque chose de ce genre:

```
fichier texte d'un entrepot avec 3 etages de produits :

0101 12-34-03   0201 99-99-99
0102 33-11-01   0202 99-99-99
0103 13-54-22   0203 99-99-99
0104 25-09-12   0204 99-99-99

0301 24-24-11   0401 24-88-11
0302 33-11-01   0402 45-24-11
0303 13-54-22   0403 24-24-11
0304 25-09-10   0404 99-99-99

3 etages de produits
4 rayons
4 etagÃ¨res par rayon
```

Enfin en dehors de cela et de ma contribution à ce rapport de soutenance je n'ai fait de contribution significative dans le projet à l'heure de cette première soutenance puisque nous avons retenu la classe graph de Thomas.

Enfin mon objectif sera maintenant de compléter la structure Graph de Thomas (notamment les listes d'adjacences, la pondération des arêtes, la création de fonctions pour simplifier la modification du graph en temps réel) avant de reprendre la suite du projet.

# Jean Loup de Beauminy:

La bibliothèque OpenGL est utilisée pour créer une visualisation 2D de la disposition du magasin. Les nœuds représentent des rayons dans le magasin, et les arcs représentent les connexions entre ces rayons.

Pour créer cette visualisation 2D, nous utilisons la sous-bibliothèque GLUT (OpenGL Utility Toolkit). GLUT permet de créer une fenêtre et de l'afficher en utilisant OpenGL. Le code utilise également la sous-bibliothèque GLU (OpenGL Utility Library) pour des fonctions d'aide comme la création de carrés.

La position de chaque nœud est déterminée par ses coordonnées X et Y, et la position des arcs est déterminée par la position de chaque nœud connecté. Les coordonnées de chaque nœud sont accessibles directement depuis la struct Node.

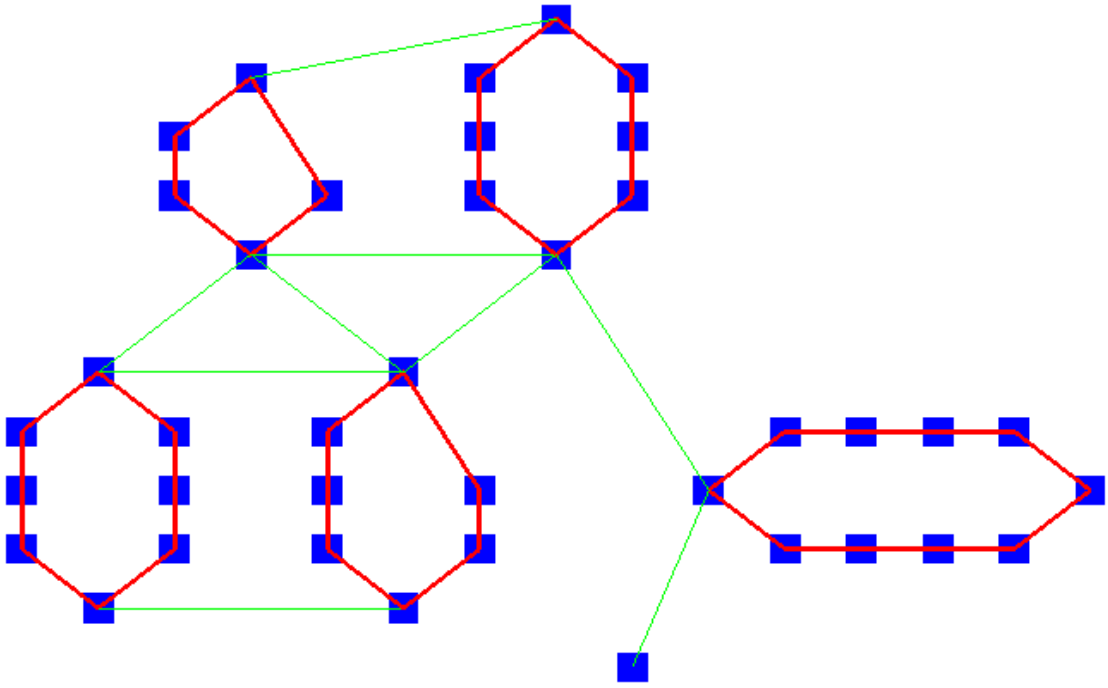
```
// lien : (node,adjNode) // Création de la ligne
drawLine(node->X * 50, node->Y * 40, adjNode->X * 50, adjNode->Y * 40);
```

Le code utilise les fonctions OpenGL pour dessiner des carrés aux emplacements des nœuds et des lignes aux emplacements des arcs..

```
// Fonction de dessin des carrés
void drawSquare(int x, int y, int size) {
     glBegin(GL_QUADS);
     glColor3f(0.0f, 0.0f, 1.0f); // Bleu
     glVertex2i(x - size, y - size);
     glVertex2i(x + size, y - size);
     glVertex2i(x + size, y + size);
     glVertex2i(x - size, y + size);
     glEnd();
}
```

```
// Fonction de dessin des traits
void drawLine(int x1, int y1, int x2, int y2) {
     glBegin(GL_LINES);
     glVertex2i(x1, y1);
     glVertex2i(x2, y2);
     glEnd();
}
```

Le code utilise différentes couleurs et épaisseurs pour les arcs, en fonction de leur type.



Pour différencier les liens verts et les rouges, j'ai créé 2 catégories de longueurs, ce qui implique le calcul de la longueur des liens.
La représentation de chaque lien a donc été faite comme ceci :

```
double distance = sqrt(pow(adjNode->X - node->X, 2) + pow(adjNode->Y -
node->Y, 2));
     if (distance >= 2.5) { // Si lien entre rayons
          glLineWidth(1); // Modifier l'épaisseur de la ligne
          glColor3f(0.0, 1.0, 0.0); // Vert pour les liens entre rayons
     }else{
          glLineWidth(5);// Modifier l'épaisseur de la ligne
          glColor3f(1.0, 0.0, 0.0); // Rouge pour les liens entre articles
     }
drawLine(node->X * 50, node->Y * 40, adjNode->X * 50, adjNode->Y * 40);
```

Nous avons rencontré plusieurs difficultés lors de la représentation graphique en C. D'abord, il y avait une certaine confusion quant à la bibliothèque graphique à utiliser. Après avoir examiné les avantages et les inconvénients de plusieurs bibliothèques, il a été décidé d'utiliser OpenGL pour sa grande flexibilité et sa compatibilité avec les systèmes d'exploitation. Plusieurs essais ont été faits avec SDL, GTK+, Qt et Allegro.

Avec OpenGL, nous avons donc pu représenter un magasin. Cette représentation minimaliste, qui, graphiquement, est apparentée à un graph, nous est très utile pour tester notre travail et visualiser ce qui se passe réellement. Cette représentation prendra tout son sens quand nous calculerons les chemin entre les produits.

# Annexes:

Rappel du lien du site:

https://perfectcourse.neocities.org/

Planning mis à jour:

| Date (plus ou moins approximative) | Tâche ou sous problème à résoudre |
|---|---|
| 30 janvier 2023: | Remise des listes des groupes |
| 31 janvier 2023: | Validation définitive des groupes |
| | Formalisation du projet et création du cahier des charges |
| 06 février 2023: | Remise du cahier des charges |
| 08 février 2023 : | Validation théorique du cahier des charges |
| 17 février 2023 | Validation effective du cahier des charges |
| Semaine du 20 février 2023: | Implémentation des graphs en C<br><br>Création de l'outil de modélisation de l'entrepôt sous forme de graphe (via l'implémentation des graphs en C),<br><br>Codage de la base du site pour héberger les fichiers de formalité (code, rapports, …) |
| Semaine du 27 février 2023: | Implémentation d'une reconnaissance d'image pour lire les qr codes. Localisation de l'employé grâce au QR code. |
| Semaine du 6 mars 2023: | Création d'une interface de démonstration dans laquelle on peut faire se déplacer l'employé et simuler le chemin emprunté,<br><br>1 ère Soutenance |

| | |
|---|---|
| Semaine du 13 mars 2023: | Implémentation du premier algorithme de recherche de chemin, basé sur la distance parcouru (et donc le temps) |
| Semaine du 20 mars 2023: | Création d'un entrepôt type et archivage d'une liste de produits imaginaires.<br>Création de l'interface permettant d'enregistrer des références |
| Semaine du 27 mars 2023: | Création de l'algorithme de recherche de chemin selon le poids et la fragilité des objets |
| Semaine du 11 avril 2023: | Semaine de rattrapage des différents retards du projet, débogage d'éventuelles coquilles et/ou rectification de la feuille de route.<br>Création du design définitif du site et mise en place de ce dernier |
| Semaine du 18 avril 2023: | Mise en place de localisation "en temps réel" grâce à une simulation de balise bluetooth (amélioration de la précision comparé au QR code) |
| Semaine du 25 avril 2023: | Détection de l'égarement de l'employé et création mise en place du correcteur d'itinéraire.<br><br>Préparation de la seconde soutenance. |
| Semaine du 1er mai 2023: | 2éme soutenance (nous supposons) |
| Semaine du 8 mai 2023: | Mise en place de l'indication "en temps réel" de la direction à suivre pour l'employé (dans la simulation toujours). |
| Semaine du 15 mai 2023: | Mise en place d'un parser pour récupérer la liste des produits selon les références données |
| Semaine du 22 mai 2023: | Mise en place d'un générateur aléatoire de commande et récupération des dites commandes |

| | |
|---|---|
| | pour tester l’application |
| à partir de maintenant les dates sont toujours à supposer puisque nous ne savons pas si seulement la première soutenance a été décalé ou non | |
| | Tests divers et variés et amélioration de l’interface graphique |
| | Semaine de rattrapage des différents retards du projet, débogage d’éventuelles coquilles.<br>Début de préparation du rapport de projet |
| | Débogage des éventuels bugs persistants, préparation du rapport de projet ainsi que de la soutenance |
| Semaine du 29 mai 2023: | Dernière soutenance (à voir si cette dernière sera décalé ou non) |

# Diverses captures d’écrans:

## Le site: