**RAPPORT PROJET PROGRAMMATION EN LANGAGE C**

[Sous-titre du document]

Réalisé par : NDOM Marie Céline & TCHAKOTE Manuel Bryan

ECOLE SUPERIEURE DES SCIENCES DE LA FINANCE, DE L’ASSURANCE ET DES RISQUES (ESSFAR)

Sous la supervision de Dr. TSOPZE Norbert

Table des matières

[INTRODUCTION 2](#_Toc30392767)

[1. LES DIFFERENTS SOUS-PROGRAMMES 3](#_Toc30392768)

[a) void saisie() 3](#_Toc30392769)

[b) void modifier(int NCIN) 4](#_Toc30392770)

[c) void supprimer(int NCIN) 4](#_Toc30392771)

[d) void admis() 5](#_Toc30392772)

[e) void attente() 6](#_Toc30392773)

[f) double statistiques(Decision dec) 7](#_Toc30392774)

[g) void supprimer\_20() 8](#_Toc30392775)

[h) void affiche\_merite() 9](#_Toc30392776)

[i) void recherche(int NCIN) 9](#_Toc30392777)

[j) void presentation() 10](#_Toc30392778)

[k) int main() 10](#_Toc30392779)

[2. LIMITES DU CODE SOURCE 12](#_Toc30392780)

# INTRODUCTION

Dans le cadre de notre formation en première année filière Mathématiques et Informatique appliquées à l’économie, à la finance et à l’assurance à l’ESSFAR, nous avons suivi un cours d’algorithmique et de programmation en C au terme duquel l’enseignant nous a confié un projet afin que nous puissions mettre en application les connaissances reçues. Par groupes de 2, il nous a été demandé de définir une dizaine de fonctions permettant d’effectuer diverses opérations en rapport avec la gestion d’un concours. Les 400 lignes de code qui nous ont aidé à la résolution de ce problème seront succinctement analysées et par la suite, nous présenterons les limites connues de notre programme.

# LES DIFFERENTS SOUS-PROGRAMMES

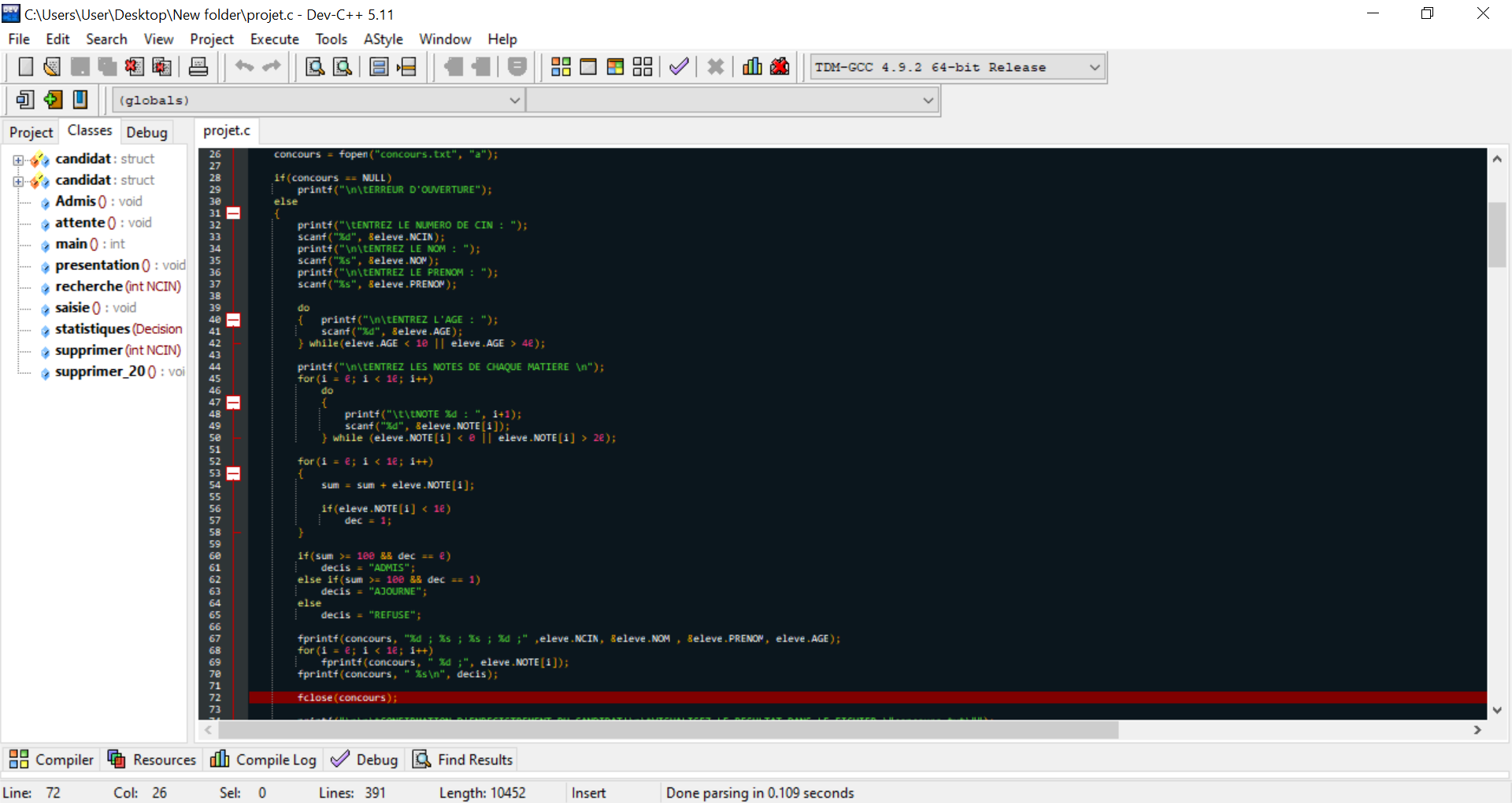
## void saisie()

Cette fonction permet de saisir et d’ajouter les informations relatives à chaque candidat dans le fichier ***concours.txt***. Chaque candidat est un enregistrement représenté par son numéro de ***CIN***, son ***NOM***, son ***PRENOM***, son ***AGE***, son tableau de 10 ***NOTES***, et la ***DECISION*** du jury qui est une énumération contenant les éléments ***ADMIS***, ***REFUSE*** et ***AJOURNE***. Cette fonction utilise les variables de type candidat, int, double et char, et nécessite l’ouverture du fichier ***concours.txt*** en mode ajout.

* ***candidat eleve*** désigne l’enregistrement caractérisant chaque concourant
* ***int i*** sert de compteur dans les boucles for pour le traitement des 10 notes
* ***double sum*** contient la somme des notes du candidat
* ***int dec*** est initialisée à 0, et prend la valeur 1 si un candidat dont la somme des notes est supérieure à 100 (le minimum pour être ***ADMIS***) a une de ses notes inférieure à 10 (la moyenne)
* ***char \*decis*** enregistre la décision du jury en fonction des notes saisies

Le traitement s’effectue de la manière suivante :

* On déclare les différentes variables, on ouvre le fichier concerné et on teste son ouverture
* Dans le cas d’une ouverture réussie, on demande à l’utilisateur d’entrer le numéro de ***CIN***, le ***NOM***, le ***PRENOM***, l’***AGE*** du candidat et enfin ses 10 ***NOTES*** ; l’AGE du candidat doit être compris entre 10 et 40 ans, et les NOTES sont des entiers pris entre 0 et 20
* On stocke ensuite dans ***sum*** la somme des notes du candidat et on vérifie s’il n’a pas eu de sous moyenne ; cela permet de compléter le champ ***DECISION***
* La collecte des données étant terminée, l’on procède à leur écriture dans le fichier ***concours.txt*** en séparant une information de l’autre par la séquence « ; »
* Un message est affiché à l’utilisateur pour confirmer l’enregistrement du candidat dans la base de données



## void modifier(int NCIN)

Nous n’avons malheureusement pas été en mesure de programmer cette fonction qui devait permettre de corriger les données d’un candidat déjà existant dans le fichier ***concours.txt***.

## void supprimer(int NCIN)

***supprimer(int NCIN)*** prend en paramètre un entier ***NCIN*** qui est le numéro de CIN du candidat que l’utilisateur souhaite retirer du fichier ***concours.txt***. Cette fonction utilise les variables de type candidat, int et char, et nécessite l’ouverture du fichier ***concours.txt*** en mode lecture et du fichier ***concours.select*** en mode ajout.

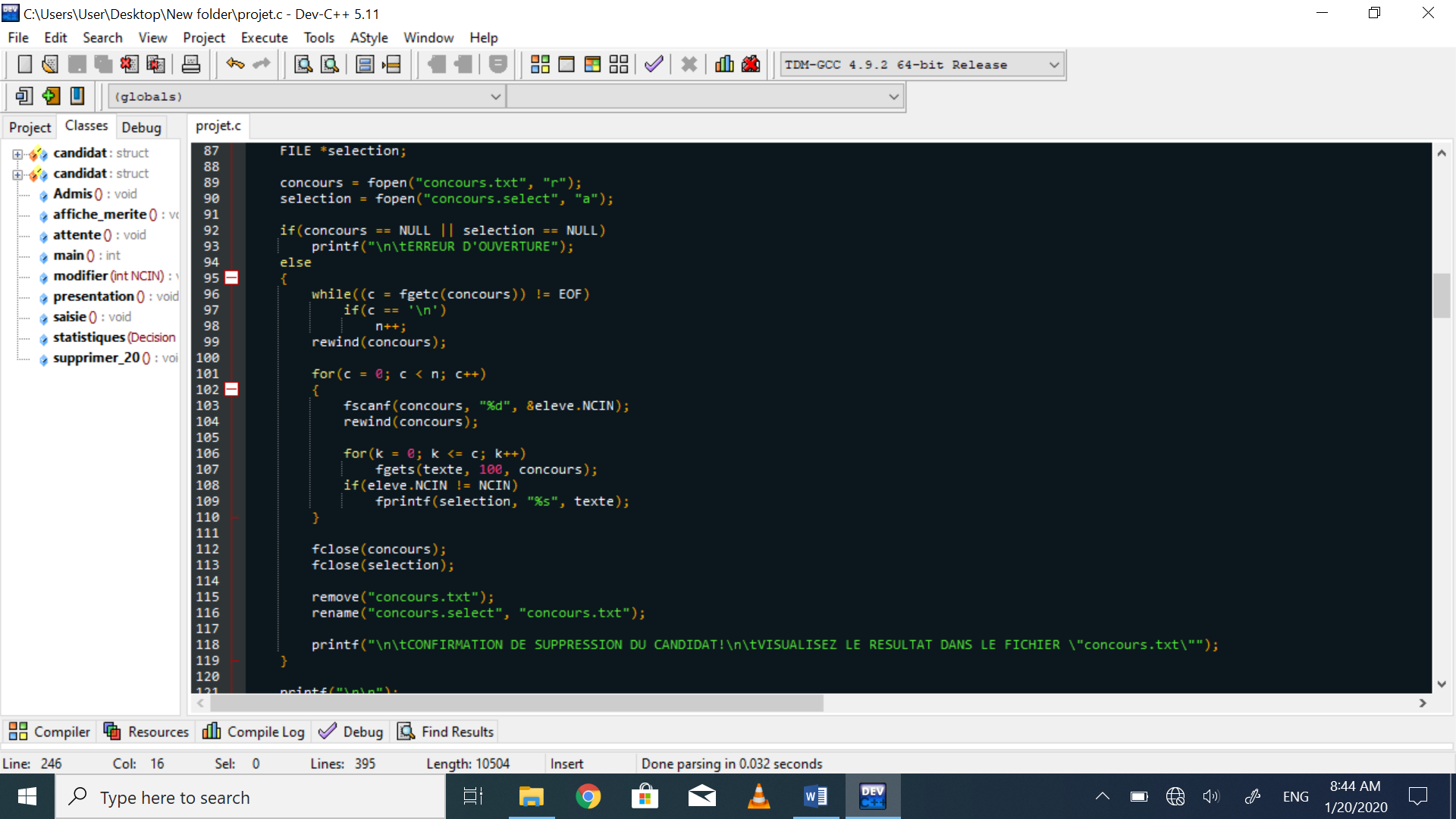
* ***candidat eleve*** désigne l’enregistrement caractérisant chaque concourant
* ***int c*** parcourt successivement tous les caractères du fichier ***concours.txt*** ; l’on s’intéresse ici au nombre de retours à la ligne
* ***int n*** enregistre le nombre de retours à la ligne et donc le nombre de candidats présents dans la base de données
* ***int k*** permet de cibler les candidats l’un après l’autre pour effectuer le traitement proprement dit
* ***char texte[100]*** parcourt successivement toutes les lignes du fichier ***admis.txt*** ; l’on s’intéresse ici au champ contenant le ***NCIN*** dans la ligne ciblée

Le traitement s’effectue de la manière suivante :

* On déclare les différentes variables, on ouvre les fichiers concernés et on teste leur ouverture
* Dans le cas d’une ouverture réussie, on stocke dans ***n*** le nombre de candidats inscrits
* On effectue ensuite la lecture d’une chaine formatée dans le fichier ***concours.txt*** ; les informations sont stockées dans la variable ***eleve.NCIN***
* Le traitement se poursuit par la lecture successive des informations relatives à chacun des candidats à l’aide de ***texte[100]***, puis vient le test sur le champ ***NCIN*** de la ligne lue
* Au fur et à mesure que l’on parcourt les lignes du fichier ***concours.txt***, le fichier ***concours.select*** se remplit des informations des candidats retenus (ceux dont le ***NCIN*** est différent du paramètre de la fonction)
* La dernière manœuvre consiste à supprimer le fichier ***concours.txt*** et à donner son nom au fichier ***concours.select***
* Le candidat ayant été supprimé de la liste des concourants, (ou pas - si le ***NCIN*** entré ne correspondait à personne), les fonctions ***admis()*** et ***attente()*** sont appelées pour mettre à jour les fichiers ***admis.txt*** et ***attente.txt***
* Un message est affiché à l’utilisateur pour confirmer l’effacement du candidat de la base de données

Etant donné que le traitement ci-dessus nécessite un passage préalable de paramètre par l’utilisateur, il est précédé d’une invitation à saisir une donnée en fonction de laquelle le paramètre sera fixé.

Etant donné que le traitement ci-dessus peut s’avérer délicat, il est précédé d’une demande de confirmation avant son exécution.



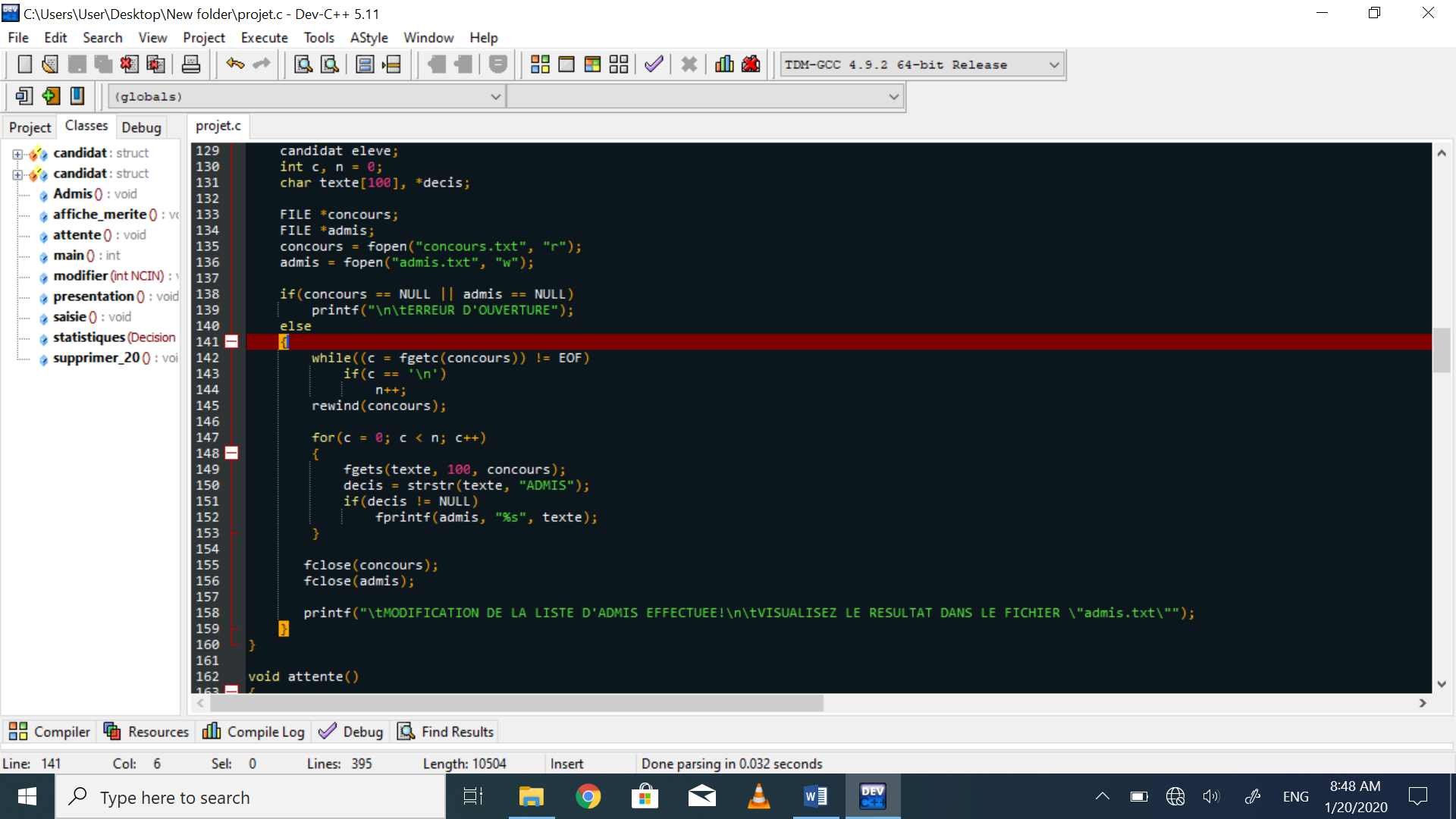
## void admis()

Le sous-programme ***admis()*** permet d’écrire dans le fichier ***admis.txt*** les informations des candidats retenus à l’issue du concours. Il utilise les variables de type candidat, int et char, et nécessite l’ouverture du fichier ***concours.txt*** en mode lecture et du fichier ***admis.txt*** en mode écriture.

* ***candidat eleve*** désigne l’enregistrement caractérisant chaque concourant
* ***int c*** parcourt successivement tous les caractères du fichier ***concours.txt***
* ***int n*** enregistre le nombre de candidats présents dans la base de données
* ***char texte[100]*** parcourt successivement toutes les lignes du fichier ***concours.txt*** ; l’on s’intéresse ici au champ contenant la ***DECISION*** du jury dans la ligne ciblée
* ***char \*decis*** permet la recherche des valeurs de l’énumération ***Decision*** (***ADMIS***, ***AJOURNE***, ou ***REFUSE***) dans la chaine ***texte[100]***

Le traitement s’effectue de la manière suivante :

* On déclare les différentes variables, on ouvre les fichiers concernés et on teste leur ouverture
* Dans le cas d’une ouverture réussie, on affecte à ***n*** le nombre de candidats inscrits et on lit tour à tour les informations relatives à chacun d’eux a l’aide de ***texte[100]***
* Pour chaque ligne lue, l’on effectue une recherche qui permet d’y détecter la présence de la chaine « ***ADMIS*** », et donc de déterminer si ***\*decis*** contient la valeur NULL ou pas
* Au fur et à mesure que l’on parcourt les lignes du fichier ***concours.txt***, le fichier ***admis.txt*** se remplit des informations des candidats pour lesquels ***\*decis*** est différent de NULL
* Un message est affiché à l’utilisateur pour confirmer la modification du fichier ***admis.txt***



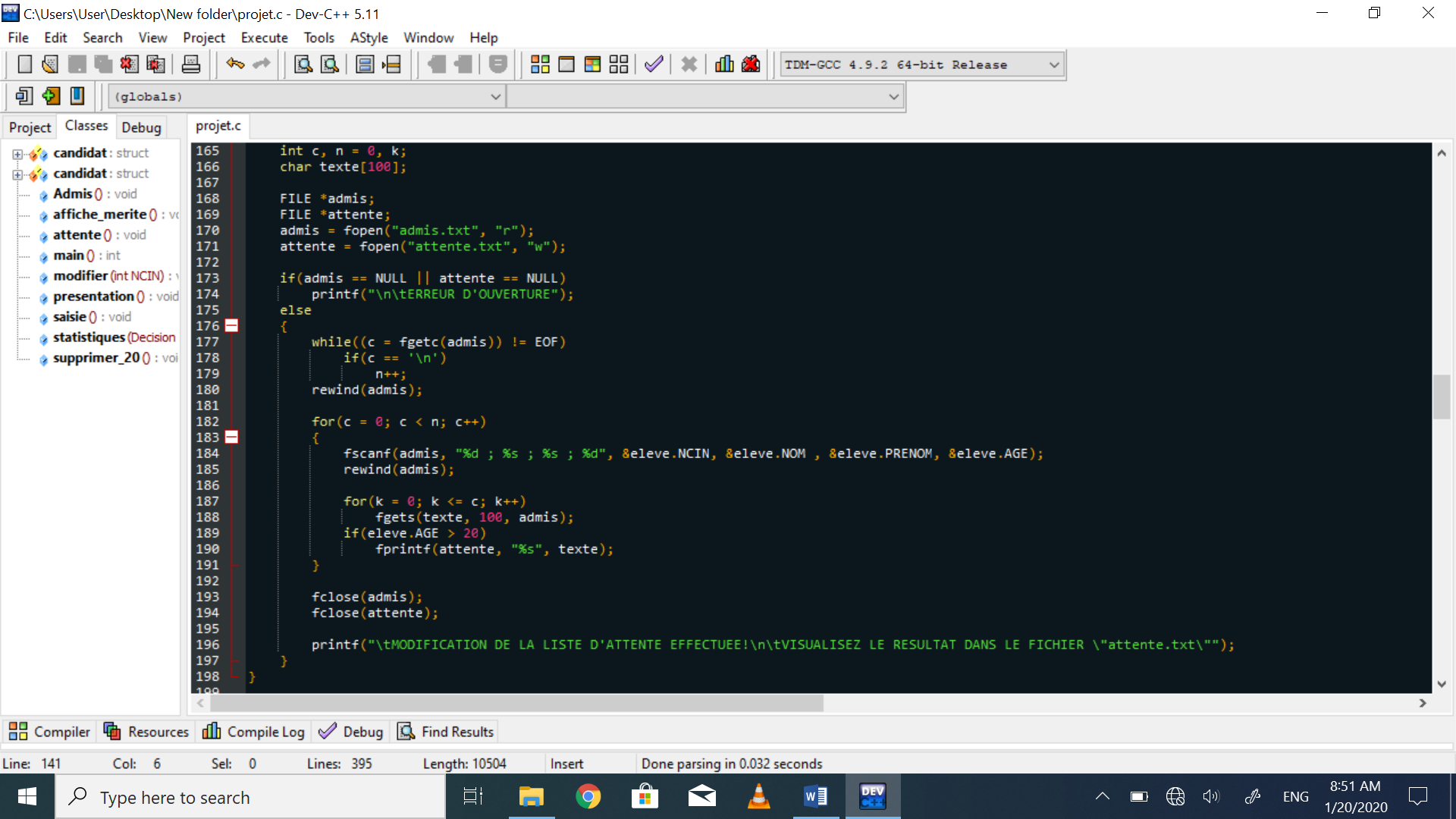
## void attente()

La procédure ***attente()*** insère dans le fichier ***attente.txt*** les informations relatives aux candidats qui sont ***ADMIS*** mais qui sont âgés de plus de 20 ans. Cette fonction utilise les variables de type candidat, int et char, et nécessite l’ouverture du fichier ***admis.txt*** en mode lecture et du fichier ***attente.txt*** en mode écriture.

* ***candidat eleve*** désigne l’enregistrement caractérisant chaque concourant
* ***int c*** parcourt successivement tous les caractères du fichier ***admis.txt***
* ***int n*** enregistre le nombre de candidats présents dans la base de données
* ***int k*** permet de cibler les candidats l’un après l’autre pour effectuer le traitement proprement dit
* ***char texte[100]*** parcourt successivement toutes les lignes du fichier ***admis.txt*** ; l’on s’intéresse ici au champ ***eleve.AGE*** dans la ligne ciblée

Le traitement s’effectue de la manière suivante :

* On déclare les différentes variables, on ouvre les fichiers concernés et on teste leur ouverture
* Dans le cas d’une ouverture réussie, on stocke dans ***n*** le nombre de candidats inscrits
* On effectue ensuite la lecture d’une chaine formatée dans le fichier ***admis.txt*** ; les informations sont stockées dans les variables ***eleve.NCIN***, ***eleve.NOM***, ***eleve.PRENOM*** et ***eleve.AGE***
* Le traitement se poursuit par la lecture successive des informations relatives à chacun des candidats à l’aide de ***texte[100]***, puis vient le test sur le champ ***AGE*** de la ligne lue
* Au fur et à mesure que l’on parcourt les lignes du fichier ***admis.txt***, le fichier ***attente.txt*** se remplit des informations des candidats dont l’âge est supérieur ou égal à 20
* Un message est affiché à l’utilisateur pour confirmer la modification du fichier ***attente.txt***



## double statistiques(Decision dec)

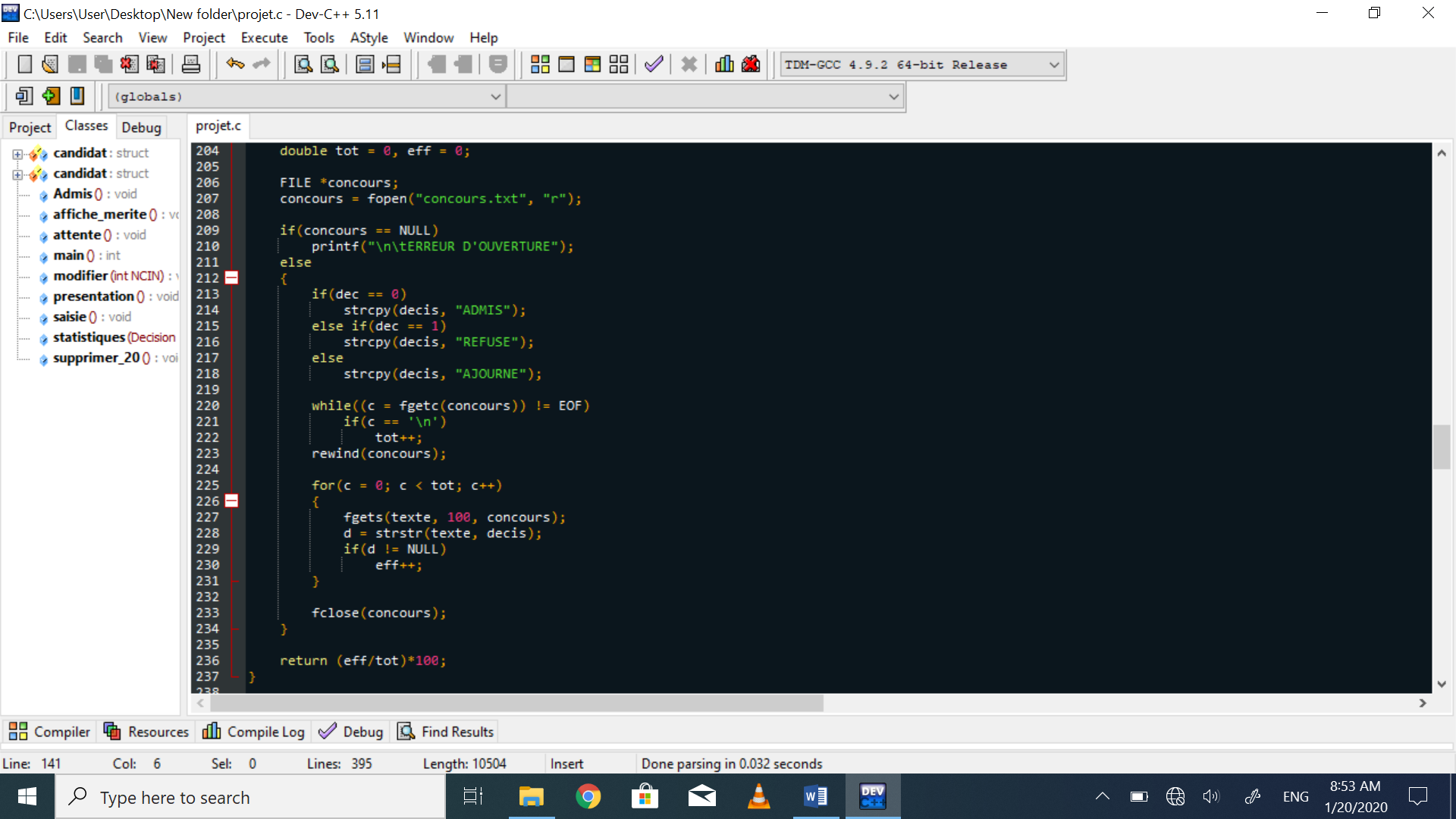
Cette fonction prend en paramètre un entier (0, 1 ou 2) qui renvoie à l’indice des éléments dans l’énumération ***Decision*** (***ADMIS***, ***REFUSE***, ***AJOURNE***), et retourne le pourcentage de candidats dont la mention correspond à l’élément ciblé. Elle utilise les variables de type int, char et double, et nécessite l’ouverture du fichier ***concours.txt*** en mode lecture.

* ***char decis[8]*** contient ***ADMIS***, ***REFUSE*** ou ***AJOURNE***, en fonction du paramètre passe à la fonction
* ***int c*** parcourt successivement tous les caractères du fichier ***concours.txt***
* ***int tot*** enregistre le nombre de candidats présents dans la base de données
* ***char*** ***texte[100]*** parcourt successivement toutes les lignes du fichier ***concours.txt*** ; l’on s’intéresse ici à la présence de la chaine contenue par ***decis[8]*** dans la ligne ciblée
* ***char \*d*** permet de confirmer la présence de la chaine contenue par ***decis[8]*** dans la ligne ciblée
* ***int eff*** calcule le nombre de candidats dont la mention correspond au paramètre passe à la fonction

Le traitement s’effectue de la manière suivante :

* On déclare les différentes variables, on ouvre le fichier concerné et on teste son ouverture
* Dans le cas d’une ouverture réussie, on affecte à la variable ***decis*** l’élément dans l’énumération ***Decision*** correspondant à la valeur en paramètre de la fonction
* On stocke ensuite dans ***tot*** le nombre de candidats inscrits et on lit tour à tour les informations relatives à chacun d’eux a l’aide de ***texte[100]***
* Pour chaque ligne lue, l’on effectue une recherche qui permet de comparer la décision du jury au représentant du paramètre de la fonction ; c’est à ce niveau que ***d*** intervient
* S’il y a correspondance, il y a incrémentation de ***eff***
* A la fin du traitement, il suffit de retourner la valeur de ***(eff/tot)\*100*** en guise de pourcentage soit d’***ADMIS***, soit de ***REFUSE***S, soit d’***AJOURNE***S

Etant donné que le traitement ci-dessus nécessite un passage préalable de paramètre par l’utilisateur, il est précédé d’une invitation à saisir une donnée en fonction de laquelle le paramètre sera fixé.



## void supprimer\_20()

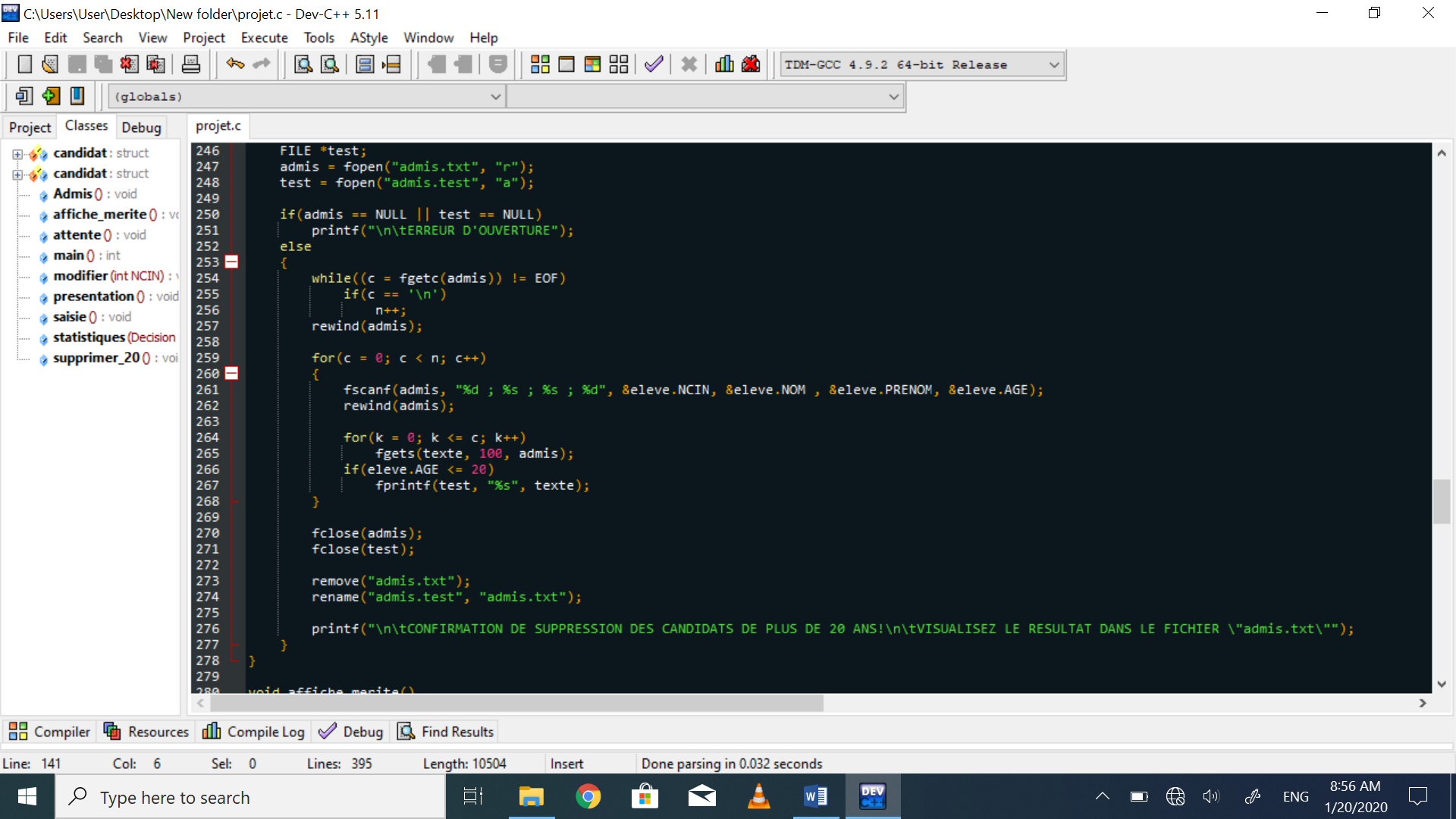
Le sous-programme ***supprimer\_20()*** retire de la liste d’***ADMIS*** les candidats dont l’âge dépasse 20 ans. Il utilise les variables de type candidat, int et char, et nécessite l’ouverture du fichier ***admis.txt*** en mode lecture, et d’un autre nomme ***admis***.***test*** en mode ajout.

* ***candidat eleve*** désigne l’enregistrement caractérisant chaque concourant
* ***int c*** parcourt successivement tous les caractères du fichier ***admis.txt***
* ***int n*** enregistre le nombre de candidats présents dans la base de données
* ***int k*** permet de cibler les candidats l’un après l’autre pour effectuer le traitement proprement dit
* ***char*** ***texte[100]*** parcourt successivement toutes les lignes du fichier ***admis.txt*** ; l’on s’intéresse ici au champ ***eleve.AGE*** dans la ligne ciblée

Le traitement s’effectue de la manière suivante :

* On déclare les différentes variables, on ouvre les fichiers concernes et on teste leur ouverture
* Dans le cas d’une ouverture réussie, on stocke dans ***n*** le nombre de candidats inscrits
* On effectue ensuite la lecture d’une chaine formatée dans le fichier ***admis.txt*** ; les informations sont stockées dans les variables ***eleve.NCIN***, ***eleve.NOM***, ***eleve.PRENOM*** et ***eleve.AGE***
* Le traitement se poursuit par la lecture successive des informations relatives à chacun des candidats à l’aide de ***texte[100]***, puis vient le test sur le champ ***AGE*** de la ligne lue
* Au fur et à mesure que l’on parcourt les lignes du fichier ***admis.txt***, le fichier ***admis***.***test*** se remplit des informations des candidats retenus (ceux dont l’âge est inférieur ou égal à 20)
* La dernière manœuvre consiste à supprimer le fichier ***admis.txt*** et à donner son nom au fichier ***admis***.***test***
* Un message est affiché à l’utilisateur pour confirmer la suppression des candidats de plus de 20 ans dans le fichier ***admis.txt***

Etant donné que le traitement ci-dessus peut s’avérer délicat, il est précédé d’une demande de confirmation avant son exécution.



## void affiche\_merite()

Nous n’avons malheureusement pas été en mesure de programmer cette fonction qui devait classer par ordre de mérite les candidats ***ADMIS*** au concours.

## void recherche(int NCIN)

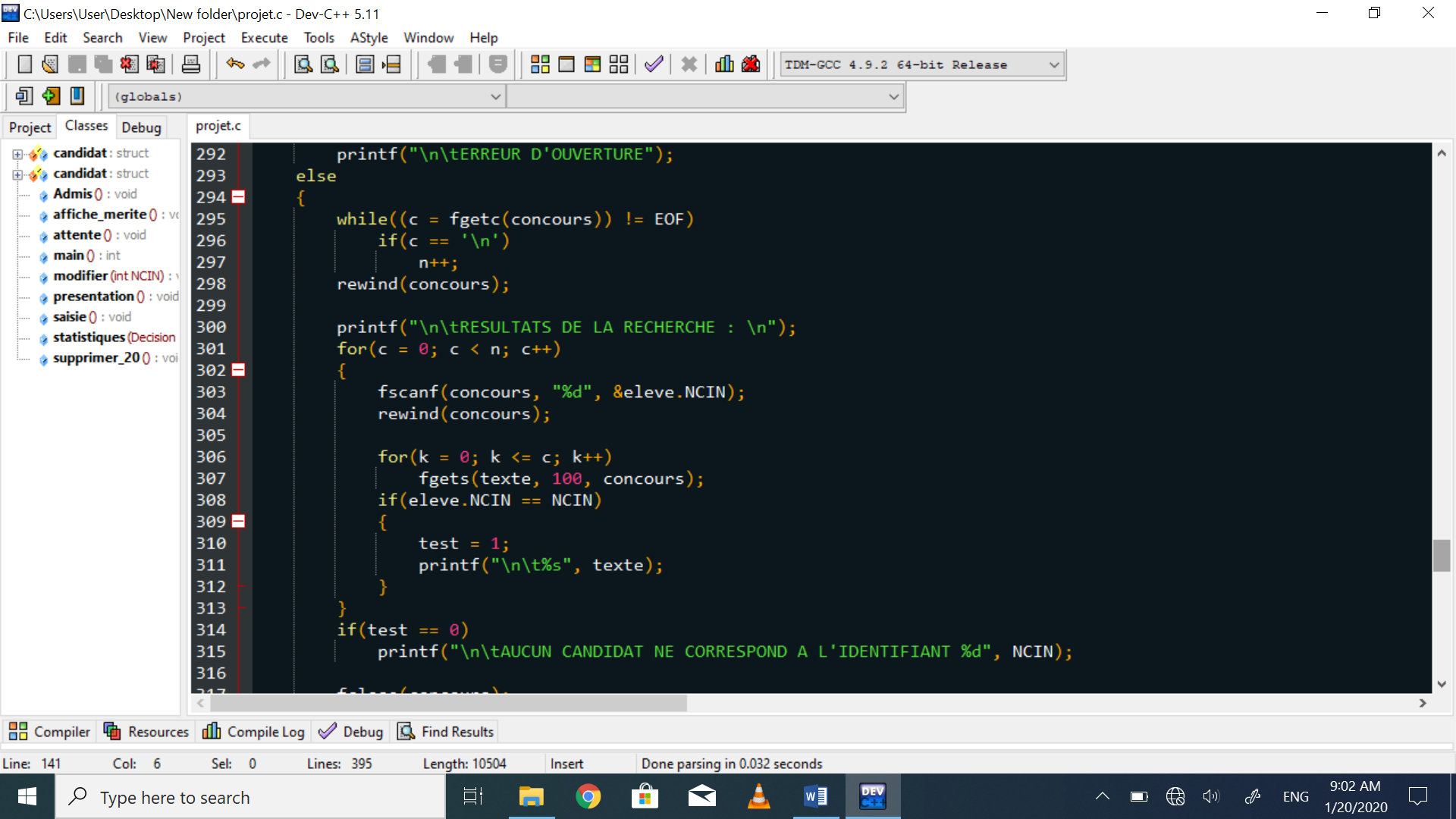
Il s’agit ici de la procédure qui permet d’afficher les données du candidat dont le ***NCIN*** est passé en paramètre. Elle utilise les variables de type candidat, int et char, et nécessite l’ouverture du fichier ***concours.txt*** en mode lecture.

* ***candidat eleve*** désigne l’enregistrement caractérisant chaque concourant
* ***int c*** parcourt successivement tous les caractères du fichier ***concours.txt***
* ***int n*** enregistre le nombre de candidats présents dans la base de données
* ***int k*** permet de cibler les candidats l’un après l’autre pour effectuer le traitement proprement dit
* ***char*** ***texte[100]*** parcourt successivement toutes les lignes du fichier ***concours.txt*** ; l’on s’intéresse ici au champ ***eleve.NCIN*** dans la ligne ciblée
* ***int test*** contient 0 par défaut, on lui affecte l’entier 1 si la valeur passée en paramètre correspond au ***NCIN*** d’un candidat

Le traitement s’effectue de la manière suivante :

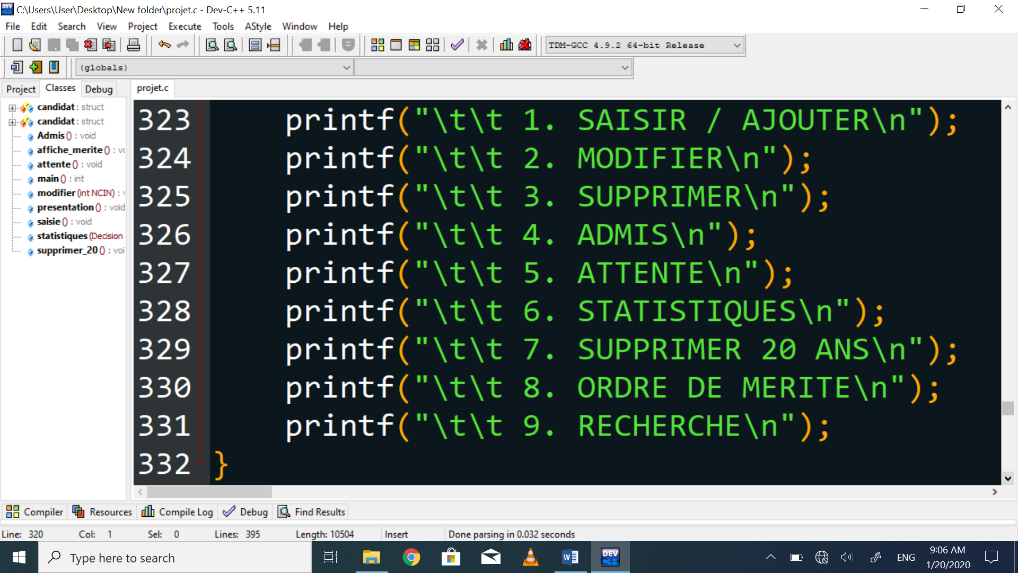
* On déclare les différentes variables, on ouvre le fichier concerné et on teste son ouverture
* Dans le cas d’une ouverture réussie, on stocke dans ***n*** le nombre de candidats inscrits
* On effectue ensuite la lecture d’une chaine formatée dans le fichier ***concours.txt***; les informations sont stockées dans la variable ***eleve.NCIN***
* Le traitement se poursuit par la lecture successive des informations relatives à chacun des candidats à l’aide de ***texte[100]***, puis vient le test sur le champ ***NCIN*** de la ligne lue
* Si un candidat correspond à l’identifiant, ses informations sont affichées à l’écran et ***test*** prend la valeur 1
* Un message indiquant qu’il n’y a pas eu de correspondance s’affiche si ***test*** n’a pas changé de valeur tout au long du traitement

Etant donné que le traitement ci-dessus nécessite un passage préalable de paramètre par l’utilisateur, il est précédé d’une invitation à saisir une donnée en fonction de laquelle le paramètre sera fixé.



## void presentation()

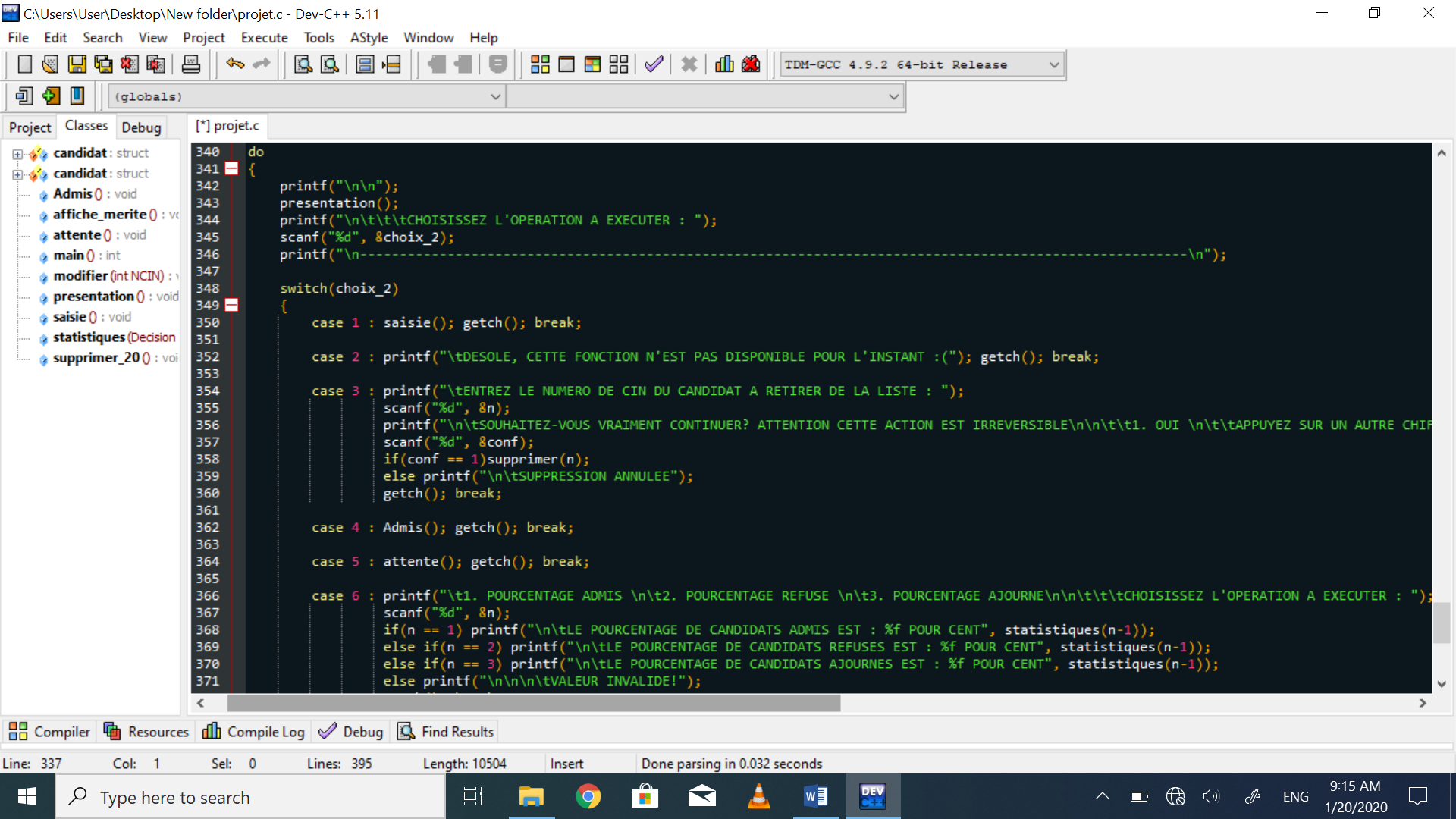
***presentation()*** gère l’affichage du menu permettant le choix des différentes actions exécutables.



## int main()

La fonction principale utilise les variables ***choix1***, ***choix2***, ***n*** et ***conf*** toutes de type int, et se compose principalement d’un switch imbriqué dans une boucle do…while.

* ***int choix\_1*** s’occupe de la fin de l’exécution du programme à travers la boucle do…while ; tant que sa valeur est 1, l’utilisateur peut effectuer des traitements concernant le concours
* ***int choix\_2*** permet de sélectionner la tâche que l’on désire effectuer dans le cadre de la gestion du concours
* ***int n*** et ***int conf*** servent à prendre des dispositions préalables à l’exécution de certaines actions. On passe par exemple ***n*** en paramètre pour la suppression et la recherche, ***n-1*** pour les différentes statistiques et la variable ***conf*** permet de confirmer les différentes actions de suppression



# LIMITES DU CODE SOURCE

Au cours des différents tests sur le programme conçu, nous avons pu noter quelques insuffisances dont la plupart sont constatées lorsqu’il y a erreur d’ouverture.

* Lorsqu’on exécute successivement les fonctions ***admis***() et ***attente()***, lorsque ***concours.txt*** n’existe pas, le message de confirmation d’exécution d’attente s’affiche quand même car ***admis()*** crée le fichier ***admis.txt*** et donc ***attente()*** peut le lire correctement.
* Pour les mêmes raisons citées ci-dessus, ***supprimer(int NCIN)*** affiche le message de confirmation d’exécution d’attente (le sous-programme fait appel à ***admis()*** puis a ***attente()***)
* ***statistiques(Decision dec)*** connait un problème similaire mais pour une raison différente ; ici, l’affichage des résultats se fait en dehors de la fonction, et donc indépendamment de l’état du fichier ***concours.txt***

En outre, nous n’avons pas pu :

* empêcher plusieurs candidats d’avoir le ***même NCIN***
* empêcher un candidat de porter comme ***NOM*** ou ***PRENOM*** les éléments ***ADMIS***, ***REFUSE*** et ***AJOURNE*** (cela pourrait contrarier les résultats de ***admis()***, ***attente()***,ou ***statistiques(Decison dec)***)
* établir une réelle distinction entre les fonctions ***saisir()*** et ***ajouter()***
* programmer les fonctions ***modifier(int NCIN)*** et ***affiche\_merite()***