Veille Technologique

Définition :

La veille technologique, c'est quoi ? L'atout principal de la veille technologique est d'identifier ou d'anticiper des innovations par secteurs d'activité. Les sources de veille technologique constituent une information stratégique incontournable pour devancer, développer et exploiter des inventions.

Ici, le sujet portera sur le remplacement du C++ par Rust.

Le C++ :

Le C++ est un langage de programmation compilé conçu par Bjarne Stroustup en 1985, permettant la programmation sous de multiples paradigmes, dont la programmation procédurale, la programmation orientée objet et la programmation générique.
Aujourd'hui, le C++ est quasiment le langage de programmation le plus puissant dans plusieurs domaines qui l'utilisent et qui sont très divers : les jeux vidéos, les services du numérique, dans l'automobile, l'aéronautique ou encore le médical.

Le Rust :

Le Rust est un langage de programmation compilé multi-paradigme conçu par Mozilla et dont la première version date de 2006. Le Rust est open source (développé par Mozilla de ce fait), ce qui sollicite l'évolution du langage par la communauté.

Developpez.com, 10/01 : C++ vs Rust : une comparaison pratique de

la vitesse de compilation et de test des deux langages de programmation

Le C++ a remporté le prix TIOBE 2022, faisant du langage C++ le langage numéro 1 de l'année, dépassant même Java. À côté, Microsoft qui utilise le C++ pour sa communauté et son contenu, utilise le tout nouveau Rust qui permet de corriger en outre les problèmes d'infrastructures que le C++ cause. Ils se complètent.
Matthew Glazar, un développeur, les compare : le Rust utilise moins de ligne de code que le C++, et est plus rapide pour les constructions complètes.
En revanche, le C++ est plus rapide pour les constructions incrémentales.

mobiskill.fr, 20/03 : Rust vs C++ : Quel stack choisir ?

En raison d'une syntaxe similaire et de meilleures approches de la gestion de la mémoire et des erreurs, Rust est aujourd'hui considéré comme une alternative à C++. Les membres de la communauté des développeurs examinent attentivement les arguments pour ou contre le passage de C++ à Rust ou inversement. Les deux langages sont compétitifs dans certains domaines comme le développement de sites Web, de jeux et d'interfaces utilisateur. Avant que vous ne décidiez définitivement lequel de ces deux langages vous allez adapter pour votre ou vos derniers projets, examinons quelques facteurs de comparaison entre Rust et C++ et leurs points forts.

Rust vs. C++ : Qualité et rapidité du codage

Le C++ est un langage à typage dynamique. Ceci explique le manque de mesures visant à prévenir les erreurs de code et à repérer les problèmes avant de compiler l'ensemble du programme. Rust est connu comme un langage à typage statique avec de nombreuses améliorations supplémentaires. C'est pourquoi son processus de validation du code est beaucoup plus rigoureux que celui du C++. Rust offre une attitude très attentive à la qualité/sécurité du code. C'est ce qui en fait l'un des principaux avantages de Rust comparé à C++. De nombreux développeurs qui écrivent leur code en Rust disent que le processus de programmation est définitivement plus confortable. Celui-ci possède une sémantique élaborée et un système bien rodé qui prend soin de la prévention des comportements indésirables. En C++, il n'est pas si facile d'éviter les comportements indéfinis, car ce langage sacrifie de telles fonctionnalités à la vitesse de travail. Néanmoins, le codage en Rust, ainsi qu'en C++, peut être délicat pour les débutants, car ce sont tous deux des langages complexes au niveau du système.
Conclusion : Rust obtient un point ici car il permet un codage plus facile.

Rust vs. C++ : Performances Rust

permet d'atteindre un niveau de performance plus élevé que C++. Il a de meilleures normes de sécurité qui réduisent le coût du processus de développement. Par exemple, pour assurer un fonctionnement plus rapide, le C++ ne dispose pas d'outils de collecte automatique des garbages, ce qui peut contribuer à de multiples erreurs d'exécution. Dans le même temps, l'une des principales différences qui rendent Rust plus sûr que C++ est que les défauts du code peuvent se traduire par des erreurs de compilation au lieu d'erreurs d'exécution. D'autre part, le C++ peut atteindre ses performances optimales et produire des applications extrêmement rapides avec moins de temps passé à la compilation et à l'exécution du code. Son énorme base de code standard (bibliothèque STL) peut couvrir de nombreuses lacunes lorsqu'elle est utilisée par un ingénieur C++ expérimenté. Lorsque vous effectuez une programmation système efficace, vous devez garder la mémoire sous contrôle de bas niveau pour éviter les fuites de mémoire coûteuses. En C++, la gestion manuelle de la mémoire et la détection des erreurs de mémoire peuvent être trop complexes ou nécessiter des efforts manuels fastidieux augmentant le coût global du développement. Rust est conçu pour minimiser de tels problèmes avec : Sa méthode d'analyse de propriété Ses vérifications de mémoire au moment de la compilation L'absence de garbage collectors « stop-the-world » Rust traite les problèmes de mémoire et examine la viabilité du code au moment de la compilation plutôt qu'au moment de l'exécution.

Conclusion :

Rust permet de créer un code plus sûr avec des coûts de développement plus faibles, mais C++ reste plus puissant. Au final, tout dépend de la puissance du compilateur, de la qualité du code, de la logique qui le sous-tend et du niveau de compétence des développeurs.