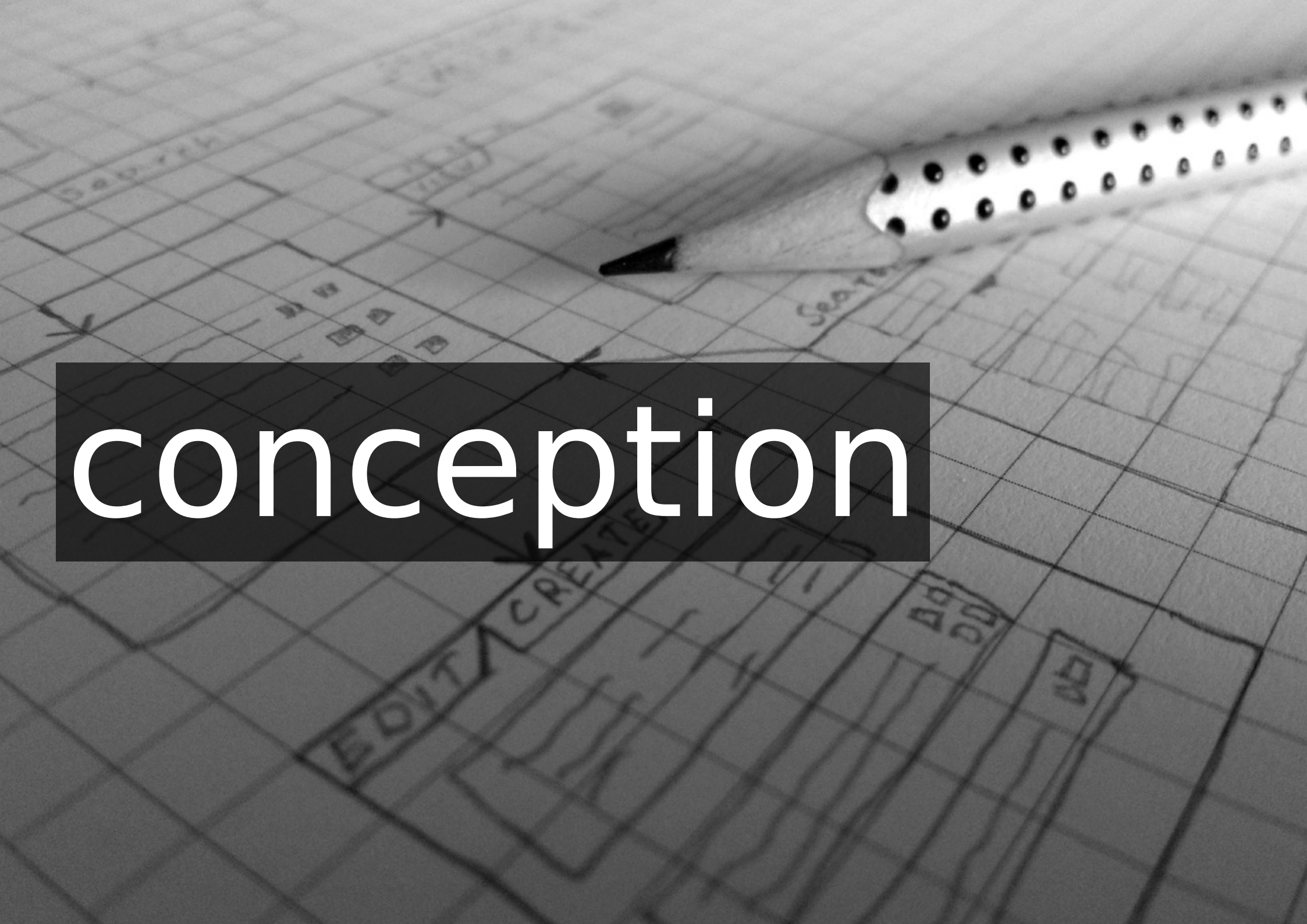
Créer une application:

le guide incomplet...
mais pratique quand même

À la base de chaque application :

idée


conception

implémentation

```
string sInput;
int iLength, iN;
double dblTemp;
bool again = true;

while (again) {
    iN = -1;
    again = false;
    getline(cin, sInput);
    system("cls");
    stringstream(sInput) >> dblTemp;
    iLength = sInput.length();
    if (iLength < 4) {
        again = true;
        continue;
    } else if (sInput[iLength - 3] != '.') {
        again = true;
        continue;
    } while (++iN < iLength) {
        if (isdigit(sInput[iN])) {
            continue;
        } else if (iN == (iLength - 3) ) {
```

```
#include <stdio.h>

int main (int argc, char *argv[])
{
    printf ("Hello, World !");
    return 0;
}
```

```
./hello-world
Hello, World !
```

Simpl(ist)e, et pourtant...

Pour créer, il faut des outils

Outils utilisés à ce stade :


• éditeur de texte
• langage de programmation
• compilateur
• système d'exploitation

Choix implicites :

1. votre plateforme favorite ♥
2. votre éditeur de texte favori ♥
3. votre langage favori ♥
4. votre compilateur/interpréteur ♥

Quelques questions à se poser avant de commencer

Quelle plate-forme(s) logicielle(s) ?

Windows ? GNU/Linux ? macOS ?
iOS ? Android ? ...
Navigateur Web ?

Quelle version(s) ?

Windows 10 ? Fedora 25 ? ...

Quelle plate-forme(s) matérielle(s) ?

x86 ? ARM ? ...

Quel langage... pour quel usage ?

contraintes différentes → choix différents

portabilité
performance
maintenance
écosystème

Quel langage... compilé ?

C, C++, Java, ...

Quel langage... interprété ?

javascript, python, perl, ...

Cible différente de l'hôte ?

→ compilation croisée
(si utilisation d'un langage compilé)

Quelle version de standard ?

C++: C++98, C+03, C++11

C: C89, C99, C11

Python: 2.7, 3.x


etc.

« Il est tentant, quand on n'a pour seul outil un marteau, de tout traiter comme un clou »

- Abraham Maslow
The Psychology of Science:
A Reconnaissance
(1966)


Trouvez vos outils
Suivez les bonnes pratiques


L'éditeur de texte

Indépendant

Vim
Emacs
SublimeText
Notepad++
etc.

Intégré dans un IDE

Eclipse
Visual Studio
Code::Blocks
GNOME Builder
etc.

Ce qu'on attend d'un éditeur

- coloration syntaxique
- marque-pages
- indentation automatique
- macros
- remplacement de texte
- bonne gestion multi-documents
- sélection rectangulaire
- extensible

Bonnes pratiques d'édition

Respectez les conventions
du document que vous éditez...

...ou adoptez les conventions du langage

Python PEP8: https://www.python.org/dev/peps/pep-0008/

Respectez l'indentation

→ configurez votre éditeur !

- facilite la lecture des diffs
- évite l'attribution
- virgule après le dernier élément

Respectez la casse

snake_case vs CamelCase vs javaCase


Trouvez votre workflow


Suivez les bonnes pratiques

LICENCE
diagramme de présentation du workflow

e, évoquer la migration C → Rust de librsvg)

compilateur/interpréteur

build systems (autotools, CMake, meson)

debogueur (gdb, nemiver; pdb) vs printf

choix d'une licence

logiciel libre et licences libres (évoquer SPDX, Creative Commons pour les resources non-code)

bonnes pratiques d'ingéniérie logiciel (joel test)

--

revision control system (Subversion, git, git-svn, cpold)

--

bugtracker (bugzilla, mantis, etc.)

--

revue de code

toolkits graphiques (Qt, GTK)

bonnes pratiques de codage (portabilité, coding style et PEP8, commentaire en anglais cf libre office, zen of python)

mauvaises pratiques (NIH, code spaghetti)

analyseur statique de code (*lint, deblint, pylint, coala)

frameworks de test unitaires

portabilité

sécurité (conteneurs, flatpak)

Crédits photo: