SAE S1.01
Implémentation d’un besoin client

Version 1 La base

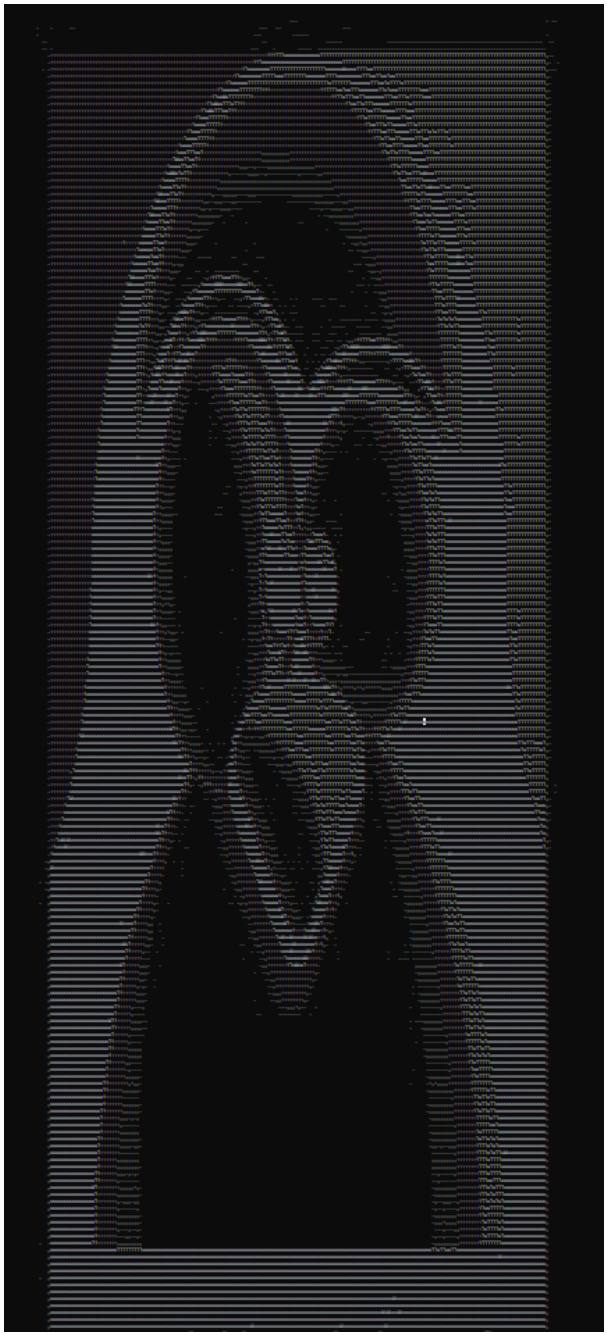
```
#include <iostream>
#ifdef _WIN32
#endif //
#include <Windows.h>
#include <fstream>
#include <sstream>
#include <vector>
#include <string>

int main()
{
#ifdef _WIN32
    SetConsoleCP(CP_UTF8);
    SetConsoleOutputCP(CP_UTF8);
#endif // _WIN32

    std::string fichier;
    std::cout << «Entrez le nom du fichier pgm : «;
    std::cin >> fichier;
    std::ifstream image(fichier,
                        std::ios_base::binary);
    std::vector<std::string> E;
    std::string entete;
    unsigned int largeur = 0;
    unsigned int hauteur = 0;
    if (!image.is_open())
    {
        std::cerr << «Problème d'ouverture du
                      fichier.\n»;
    }
    else
    {
        for (size_t i = 0; i < 4; i++)
        {
            std::getline(image, entete);
            E.push_back(entete);
            if (i == 2)
            {
                std::stringstream sstr(entete);
                sstr >> largeur >> hauteur;
            }
        }
    }
    for(size_t i = 0; i < E.size(); i++)
    {
        std::cout << E[i] << «\n»;
    }
    for (size_t j = 0; j < hauteur; j++)
    {
        std::vector<char> donnees(largeur);
        image.read(donnees.data(), largeur);
        for (size_t k = 0; k < donnees.size(); k++)
        {
            unsigned int don = donnees[k];
            while (don > 255)
            {
                don = don % 256;
            }
            if (don >= 0 && don < 32)
                std::cout << «W»;
            if (don > 31 && don < 64)
                std::cout << «w»;
            if (don > 63 && don < 96)
                std::cout << «l»;
            if (don > 95 && don < 128)
                std::cout << «i»;
            if (don > 127 && don < 160)
                std::cout << «:»;
            if (don > 159 && don < 192)
                std::cout << «,»;
            if (don > 191 && don < 224)
                std::cout << «.»;
            if (don > 223 && don < 256)
                std::cout << « «;
        }
        std::cout << '\n';
    }
}
```

Resultat

```
Entrez le nom du fichier pgm : linuxmancho.pgm
P5
# Converted from linuxmancho.jpg by img2pgm.
256 256
255
```



Version 2 Enregistrement de l’Ascii Art

```
#include <iostream>
#ifdef _WIN32
#endif //
#include <Windows.h>
#include <fstream>
#include <sstream>
#include <vector>
#include <string>

int main()
{
#ifdef _WIN32
    SetConsoleCP(CP_UTF8);
    SetConsoleOutputCP(CP_UTF8);
#endif // _WIN32

    std::string fichier;
    std::cout << «Entrez le nom du fichier pgm : »;
    std::cin >> fichier;
    std::string fichier_texte;
    std::cout << «Entrez le nom du fichier texte :»;
    std::cin >> fichier_texte;
    std::ifstream image(fichier, std::ios_
base::binary);
    std::ofstream out(fichier_texte);
    std::vector<std::string> E;
    std::string entete;
    unsigned int largeur = 0;
    unsigned int hauteur = 0;
    if (!image.is_open())
    {
        std::cerr << «Problème d’ouverture du
fichier.\n»;
    }
    else
    {
        for (size_t i = 0; i < 4; i++)
        {
            std::getline(image, entete);
            E.push_back(entete);
            if (i == 2)
            {
                std::stringstream sstr(entete);
                sstr >> largeur >> hauteur;
            }
        }
    }
    for(size_t i = 0; i < E.size(); i++)
    {
        std::cout << E[i] << «\n»;
    }
    for (size_t j = 0; j < hauteur; j++)
    {
        std::vector<char> donnees(largeur);
        image.read(donnees.data(), largeur);
        for (size_t k = 0; k < donnees.size(); k++)
        {
            unsigned int don = donnees[k];
            while (don > 255)
            {
                don = don % 256;
            }
            if (don >= 0 && don < 32)
                out << «W»;
            if (don > 31 && don < 64)
                out << «w»;
            if (don > 63 && don < 96)
                out << «l»;
            if (don > 95 && don < 128)
                out << «i»;
            if (don > 127 && don < 160)
                out << «:»;
            if (don > 159 && don < 192)
                out << «,»;
            if (don > 191 && don < 224)
                out << «.»;
            if (don > 223 && don < 256)
                out << « »;
        }
        out << ‘\n’;
    }
}
```

Version 3 Choix de la palette

console.h

```
#pragma once
#include <string>

std::string entree_nom_fichier();
std::string entree_nom_texte();
int entree_choix();
void conversion(std::string fichier, std::string
fichier_texte, std::array<std::string, 8> tab);
```

console.cpp

```
#include <iostream>
#include <fstream>
#include <vector>
#include <sstream>
#include <string>
#include <array>
#include «console.h»

std::string entree_nom_fichier()
{
    std::string fichier;
    std::cout << «Entrez le nom du fichier pgm : »;
    std::cin >> fichier;
    return fichier;
}
std::string entree_nom_texte()
{
    std::string fichier_texte;
    std::cout << «Entrez le nom du fichier texte :»;
    std::cin >> fichier_texte;
    return fichier_texte;
}
int entree_choix()
{
       int choix;
       std::cout << «Veuillez choisir votre
palette de couleur \n»
              «1 : palette\n»
              «2 : palette2\n»
              «3 : paletteUTF8\n»
              «4 : palette2UTF8\n»;
       std::cin >> choix;
       return choix;
}
```

```
void conversion(std::string fichier,
                std::string fichier_texte,
                std::array<std::string, 8> tab)
{
    std::ifstream image(fichier,
                        std::ios_base::binary);
    std::ofstream out(fichier_texte);
    std::vector<std::string> E;
    std::string entete;
    unsigned int largeur = 0;
    unsigned int hauteur = 0;
    if (!image.is_open())
        std::cerr << «Problème d’ouverture du
                     fichier.\n»;
    else
    {
        for (size_t i = 0; i < 4; i++)
        {
            std::getline(image, entete);
            E.push_back(entete);
            if (i == 2)
            {
                std::stringstream sstr(entete);
                sstr >> largeur >> hauteur;
            }
        }
    }
    for (size_t i = 0; i < E.size(); i++)
    {
        std::cout << E[i] << «\n»;
    }
    for (size_t j = 0; j < hauteur; j++)
    {
        std::vector<char> donnees(largeur);
        image.read(donnees.data(), largeur);
        for (size_t k = 0; k < donnees.size(); k++)
        {
            unsigned int don = donnees[k];
            while (don > 255)
            {
                don = don % 256;
            }
            if (don >= 0 && don < 32)
                out << tab[0];
            if (don > 31 && don < 64)
                out << tab[1];
            if (don > 63 && don < 96)
                out << tab[2];
            if (don > 95 && don < 128)
                out << tab[3];
            if (don > 127 && don < 160)
                out << tab[4];
            if (don > 159 && don < 192)
                out << tab[5];
            if (don > 191 && don < 224)
                out << tab[6];
            if (don > 223 && don < 256)
                out << tab[7];
        }
        out << ‘\n’;
    }
}
```

fonction.h

```
#pragma once
#include <fstream>
#include <string>
#include <array>

std::array<std::string, 8> palette(int choix);
```

fonction.cpp

```
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include <array>
#include «fonction.h»

std::array<std::string, 8> palette(int choix)
{
    std::string couleur;
    std::array<std::string, 8> tab;
    switch (choix)
    {
    case 1:
    {
        std::ifstream palette(«palette.txt»);
        for (size_t i = 0; i < tab.size(); i++)
        {
            std::getline(palette, couleur);
            tab[i] = couleur;
        }
        break;
    }
    case 2:
    {
        std::ifstream palette(«palette2.txt»);
        for (size_t i = 0; i < tab.size(); i++)
        {
            std::getline(palette, couleur);
            tab[i] = couleur;
        }
        break;
    }
    case 3:
    {
        std::ifstream palette(«paletteUTF8.txt»);
        for (size_t i = 0; i < tab.size(); i++)
        {
            std::getline(palette, couleur);
            tab[i] = couleur;
        }
        break;
    }
    case 4:
    {
        std::ifstream palette(«palette2UTF8.txt»);
        for (size_t i = 0; i < tab.size(); i++)
        {
            std::getline(palette, couleur);
            tab[i] = couleur;
        }
        break;
    }
    }
    return tab;
}
```

solution.cpp

```
#include <iostream>
#ifdef _WIN32
#endif //
#include <Windows.h>
#include <fstream>

#include <vector>
#include <string>
#include <array>
#include «console.h»
#include «fonction.h»

int main()
{
#ifdef _WIN32
    SetConsoleCP(CP_UTF8);
    SetConsoleOutputCP(CP_UTF8);
#endif // _WIN32
        std::string fichier = entree_nom_fichier();
    std::string fichier_texte = entree_nom_texte();
    std::ifstream image(fichier, std::ios_
base::binary);
    conversion (fichier, fichier_texte,
palette(entree_choix()));
}
```

Version 4 Arguments de la ligne de commande

console.h
```
#pragma once
#include <string>

void conversion(std::string fichier, std::string fichier_texte,
std::vector<std::string> tab);
```

console.cpp
```
#include <iostream>
#include <fstream>
#include <vector>
#include <sstream>
#include <string>
#include «console.h»


void conversion(std::string fichier, std::string fichier_texte,
std::vector<std::string> tab)
{
    std::ifstream image(fichier, std::ios_base::binary);
    std::ofstream out(fichier_texte);
    std::vector<std::string> E;
    std::string entete;
    unsigned int largeur = 0;
    unsigned int hauteur = 0;
    if (!image.is_open())
    {
        std::cerr << «Problème d’ouverture du fichier.\n»;
    }
    else
    {
        for (size_t i = 0; i < 4; i++)
        {
            std::getline(image, entete);
            E.push_back(entete);
            if (i == 2)
            {
                std::stringstream sstr(entete);
                sstr >> largeur >> hauteur;
            }
        }
    }
    for (size_t i = 0; i < E.size(); i++)
    {
        std::cout << E[i] << «\n»;
    }
    for (size_t j = 0; j < hauteur; j++)
    {
        std::vector<char> donnees(largeur);
        image.read(donnees.data(), largeur);
        for (size_t k = 0; k < donnees.size(); k++)
        {
            unsigned int don = donnees[k];
            if (don > 255)
                don = don % 256;
            int intervalle = 256 / tab.size();
            int nb = don / intervalle;
            out << tab[nb];
        }
        out << ‘\n’;
    }
}
```

fonction.h
```
#pragma once
#include <string>
#include <vector>

std::vector<std::string> palette(std::string nom);
```

fonction.cpp
```
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include «fonction.h»

std::vector<std::string> palette(std::string nom)
{
    std::string couleur;
    std::vector<std::string> tab;
    std::ifstream palette(nom);
    while (!palette.eof())
    {
        std::getline(palette, couleur);
        tab.push_back(couleur);
    }
    return tab;
}
```

Cette version ne fonctionne pas non plus mais nous ne savons pas pourquoi.

Solution.cpp

```
#include <iostream>
#ifdef _WIN32
#endif //
#include <Windows.h>
#include <fstream>
#include <vector>
#include <string>
#include <array>
#include <iomanip>
#include «console.h»
#include «fonction.h»
int main(int argc, char* argv[])
{
#ifdef _WIN32
    SetConsoleCP(CP_UTF8);
    SetConsoleOutputCP(CP_UTF8);
#endif // _WIN32
    std::string fichier;
    std::string fichier_texte;
    std::string nom_palette = «palette.txt»;
       std::vector<std::string> argument = { «--input», «--output»,
«--palette», «--help» , «--width», «--height» };
       for (int i = 0; i < argc; ++i)
       {
              if (argv[i] == argument[0])
              {
                     if (argv[i + 1] != argument[1])

                            fichier = argv[i + 1];

                     else
                     {
                            std::cout << «Veuillez entre le nom de votre
fichier(.pgm) : \n»;
                            std::cin >> fichier;
                     }
              }
              if (argv[i] == argument[1])
              {
                     if (argv[i + 1] != argument[2])

                            fichier_texte = argv[i + 1];
                     else
                     {
                            std::cout << «Veuillez entre le nom de votre
fichier texte(.txt) : \n»;
                            std::cin >> fichier_texte;
                     }
              }
              if (argv[i] == argument[2])
              {
                     if (argv[i + 1] != argument[3])

                            nom_palette = argv[i + 1];
                     else
                            nom_palette = «palette.txt»;
              }
              if (argv[i] == argument[3])
              {
                     std::cout << «Usage :\nSAE_Impl.exe [options]\n\nOptions
:\n»;
                     std::cout << «--input fichier» << std::setw(15) << « «
                            << «Spécifie le fichier image à convertir\n»;
                     std::cout << std::setw(30) << « «
                            << «Si ce paramètre n’est pas spécifié, le fichier
est de-mandé via la console.\n»;
                     std::cout << «--output fichier» << std::setw(14) << « «
                            << «Spécifie le nom du fichier texte qui
contiendra l’Ascii Art.\n»;
                     std::cout << std::setw(30) << « «
                            << «Si ce paramètre n’est pas spécifié, l’Ascii
Art est sortie dans la console.\n»;
                     std::cout << «--palette fichier» << std::setw(13) << « «
                            << «Spécifie un fichier texte contenant la palette
de cou-leur Ascii.\n»;
                     std::cout << std::setw(30) << « «
                            << «Chaque ligne du fichier contient un
charactère en UTF - 8, du plus sombre au plus clair.\n»;
                     std::cout << std::setw(30) << « «
                            << «Si ce paramètre n’est pas spécifié, la
palette par dé-faut est \»W\», \»w\», \»l\», \»i\», \»:\», \»,\», \».\», \»
\» \n»;
                     std::cout << «--help» << std::setw(24) << « «
                            << «Affiche cette aide.»;
                     return 0;
              }
       }
    std::ifstream image(fichier, std::ios_base::binary);
    conversion (fichier, fichier_texte, palette(nom_palette));
}
```

```
PS D:\Users\levigne\Desktop\Solution_SAE_S1.01\Debug> & '.\SAE S1.01.exe' --help
Usage :
SAE_Impl.exe [options]

Options :
--input fichier             Spécifie le fichier image à convertir
                            Si ce paramètre n'est pas spécifié, le fichier est demandé via la console.
--output fichier            Spécifie le nom du fichier texte qui contiendra l'Ascii Art.
                            Si ce paramètre n'est pas spécifié, l'Ascii Art est sortie dans la console.
--palette fichier           Spécifie un fichier texte contenant la palette de couleur Ascii.
                            Chaque ligne du fichier contient un charactère en UTF - 8, du plus sombre au plus clair.
                            Si ce paramètre n'est pas spécifié, la palette par défaut est "W", "w", "l", "i", ":", ",
", ".", " "
--help                      Affiche cette aide.
```

Version 5 Réduction de la taille des images

console.h

```
#pragma once
#include <string>

void conversion(std::string fichier, std::string
fichier_texte, std::vector<std::string> tab, int
hauteur_max, int largeur_max);
```

console.cpp

```
#include <iostream>
#include <fstream>
#include <vector>
#include <sstream>
#include <string>
#include «console.h»


void conversion(std::string fichier, std::string
fichier_texte, std::vector<std::string> tab, int
hauteur_max, int largeur_max)
{
    std::ifstream image(fichier, std::ios_
base::binary);
    std::ofstream out(fichier_texte);
    std::vector<std::string> E;
    std::string entete;
    std::vector<std::vector<char>> tablo;
    unsigned int largeur = 0;
    unsigned int hauteur = 0;
    if (!image.is_open())
    {
        std::cerr << «Problème d’ouverture du
fichier.\n»;
    }
    else
    {
        for (size_t i = 0; i < 4; i++)
        {
            std::getline(image, entete);
            E.push_back(entete);
            if (i == 2)
            {
                std::stringstream sstr(entete);
                sstr >> largeur >> hauteur;
            }
        }
        }
        int hmax, lmax;
        if (hauteur_max == 1)
            hmax = 1;
        else
            hmax = hauteur / hauteur_max;
        if (largeur_max == 1)
            lmax = 1;
        else
            lmax = largeur / largeur_max;

        for (size_t i = 0; i < E.size(); i++)
        {
            std::cout << E[i] << «\n»;
        }
        for (size_t i = 0; i < hauteur; i += hmax)
        {
            for (size_t j = 0; j < hmax; j++)
            {
                std::vector<char> donnees(largeur);
                image.read(donnees.data(), largeur);
                tablo.push_back(donnees);
            }
    //stockage de h lignes ds un tableau

            for (size_t k = 0; k < largeur; k += lmax)
            {
                unsigned int moy = 0;
                for (size_t m = 0; m < hmax; m++)
                {
                    for (size_t n = 0; n < lmax; n++)
                    {
                         moy += (int)tablo[m][n];
                    }
                    moy /= lmax;
                }
                moy /= hmax;
                if (moy > 255)
                    moy = moy % 256;
                int intervalle = 256 / tab.size();
    //définition de la répartition plage de couleur
                int nb = moy / intervalle;      //plage
    de couleur
                out << tab[nb];
            }
            out << ‘\n’;
        }
}
```

fonction.h

```
#pragma once
#include <string>
#include <vector>

std::vector<std::string> palette(std::string nom);
```

fonction.cpp

```
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
#include «fonction.h»

std::vector<std::string> palette(std::string nom)
{
    std::string couleur;
    std::vector<std::string> tab;
    std::ifstream palette(nom);
    while (!palette.eof())
    {
        std::getline(palette, couleur);
        tab.push_back(couleur);
    }
    return tab;
}
```

Solution.cpp

```
#include <iostream>
#ifdef _WIN32
#endif //
#include <Windows.h>
#include <fstream>
#include <vector>
#include <string>
#include <array>
#include <iomanip>
#include «console.h»
#include «fonction.h»
```

```
int main(int argc, char* argv[])
{
#ifdef _WIN32
    SetConsoleCP(CP_UTF8);
    SetConsoleOutputCP(CP_UTF8);
#endif // _WIN32
    std::string fichier;
    std::string fichier_texte;
    std::string nom_palette = «palette.txt»;
    std::string largeur_max;
    std::string hauteur_max;
    std::vector<std::string> argument = { «--input», «--output», «--palette», «--help» , «--width», «--height» };
       for (int i = 0; i < argc; ++i)
       {
              if (argv[i] == argument[0])
              {
                     if (argv[i + 1] != argument[1])
                            fichier = argv[i + 1];
                     else
                     {
                            std::cout << «Veuillez entre le nom de votre fichier(.pgm) : \n»;
                            std::cin >> fichier;
                     }
              }
              if (argv[i] == argument[1])
              {
                     if (argv[i + 1] != argument[2])
                            fichier_texte = argv[i + 1];
                     else
                            fichier_texte = «rien»;
              }
              if (argv[i] == argument[2])
              {
                     if (argv[i + 1] != argument[3])
                            nom_palette = argv[i + 1];
              }
              if (argv[i] == argument[3])
              {
                     std::cout << «Usage :\nSAE_Impl.exe [options]\n\nOptions :\n»;
                     std::cout << «--input fichier» << std::setw(15) << « «
                            << «Spécifie le fichier image à convertir\n»;
                     std::cout << std::setw(30) << « «
                            << «Si ce paramètre n’est pas spécifié, le fichier est demandé via la console.\n»;
                     std::cout << «--output fichier» << std::setw(14) << « «
                            << «Spécifie le nom du fichier texte qui contiendra l’Ascii Art.\n»;
                     std::cout << std::setw(30) << « «
                            << «Si ce paramètre n’est pas spécifié, l’Ascii Art est sortie dans la console.\n»;
                     std::cout << «--palette fichier» << std::setw(13) << « «
                            << «Spécifie un fichier texte contenant la palette de couleur Ascii.\n»;
                     std::cout << std::setw(30) << « «
                            << «Chaque ligne du fichier contient un charactère en UTF - 8, du plus sombre au plus clair.\n»;
                     std::cout << std::setw(30) << « «
                            << «Si ce paramètre n’est pas spécifié, la palette par défaut est \»W\», \»w\», \»l\», \»i\», \»:\», \»,\», \».\», \» \» \n»;
                     std::cout << «--help» << std::setw(24) << « «
                            << «Affiche cette aide.»;
                     return 0;
              }
       }
       for (int o = 0; o < argc; o++)
       {
              std::cout << o << « : « << argv[o] << ‘\n’;
              if (o == 1)
                     fichier = argv[o];
              if (o == 2)
                     fichier_texte = argv[o];
              if (o == 3)
                     nom_palette = argv[o];
              else
                     nom_palette = «palette.txt»;
              if (o == 4)
                     largeur_max = argv[o];
              else
                     largeur_max = «1»;
              if (o == 5)
                     hauteur_max = argv[o];
              else
                     hauteur_max = «1»;
       }
    int hmax = std::stod(hauteur_max);
    int lmax = std::stod(largeur_max);
    std::ifstream image(fichier, std::ios_base::binary);
    conversion (fichier, fichier_texte, palette(nom_palette),hmax, lmax);
}
```

Le programme version 5 compile, le fichier txt est créé mais rien n’est écrit dessus. Nous n’avons malheureusement pas eu le temps de le terminer. Vous trouverez donc sur Visual Studio la version 4 fonctionnelle.