C++

#include <string>	Pour les strings
#include <cmath>	Pour les maths avancé
#include <vector>	Pour les vecteurs
#include <fstream>	Pour la lecture/écriture de fichier
#include <ctime> #include <cstdlib>	Pour l'aléatoire (les deux obligatoires)
#include "math.h"	Entre guillemet quand c'est un fichier que l'on a créé nous même

cout << "texte" << endl	Affiche texte dans la console
cout << "Votre age est : " << ageUtilisateur << endl;	Affiche "votre age est :" et le contenu de la variable ageUtilisateur
cin >> ageUtilisateur;	L'utilisateur entre la valeur de la variable ageUtilisateur
cin.ignore(); getline(cin, nomUtilisateur);	Mettre un cin.ignorer() **avant les getline** pour éviter les problèmes. L'utilisateur entre la valeur de la variable nomUtilisateur **(seulement pour les string)**

Variables :

bool char int unsigned int double string std::string	Booléen Un caractère. Nombre entier. Nombre entier positif ou nul. Nombre à virgule. Chaîne de caractères **Pour les fichier.h string**
int ageUtilisateur(16);	Variable int ageUtilisateur de valeur 16
int& maVariable(ageUtilisateur);	maVariable aura toujours la même valeur que ageUtilisateur **(l'inverse est aussi vrai)**
+ += - -= * *= / /= % %=	Addition Soustaction Multiplication Division Modulo

Aleatoire :

srand(time(0));	Une seule fois au début du programme
rand() % 5;	Renvoit un nombre aléatoire entre 0 et 4

Classes :

class Personnage
{
private:
 	
int coucou;
 	
public:
 	
Personnage()
{
this->coucou = 8;
}
 	
Personnage(int coucou)
{
this->coucou = coucou;
}
int getCoucou()
{
return this->coucou;
}
 	
};
	Classe personnage
	
	Attributs privés
	
	Entier coucou : privé
	
	Attributs publics
	
	Constructeur par défaut
	
	Coucou prend la valeur 8
	
	Constructeur avec paramètres
	
	Coucou = paramètre donnée
	
	getCoucou
	
	retourne coucou

class Guerrier : public Personnage
{
 	
};
	Classe guerrier hérite de personnage

cmath :

sqrt(valeur)	Racine carré
pow(valeur,puissance)	Puissances
sin(x)	Sinus
cos(x)	Cosinus
tan(x)	Tangente
exp()	Exponentielle
log()	Logarithme népérien
Log10()	Logarithme en base 10
fabs()	Valeur absolue
floor()	⌊×⌋
ceil()	⌈×⌉

Conditions:

If(condition)
{
 	
}
else if(condition)
{
 	
}
else
{
 	
}
	SI
	
	SINON SI
	
	SINON

switch (nbEnfants)
{
case 0:
CODE
break;
 	
case 1:
CODE
break;
 	
case 2:
CODE
break;
 	
default:
CODE
break;
}
	Pour la variable nbEnfants
	
	SI nbEnfants == 0
	
	STOP
	
	SINON SI nbEnfants == 1
	
	STOP
	
	SINON SI nbEnfant == 2
	
	STOP
	
	SINON
	
	STOP

&& || !	ET OU NON

while (condition)
{
 	
}
	Boucle TANT QUE

for (i=0 ; i<100 ; i++)
{
 	
}
	Boucle POUR

Fonctions :

int AjouteDeux(int nombreRecu)
{
  return valeur;
}
	Cree une fonction ajouteDeux avec l'entier nombreRecu en paramètre qui devra retourner un entier **(int au debut)**
	Retourne valeur

ajouteDeux(a);	Appelle la fonction AjouteDeux avec la variable a en paramètre

Tableaux:

int meilleurScore[5];	Cree un tableau de int de 5 valeur
meilleursScores[0] = 118218;	Modifie la première valeur du tableau par 118218
variableString[0]	Les strings sont configurable comme un tableau **(ici, première lettre du string)**
texte.size()	Renvoie la taille d'une chaine de caractère

Listes:

vector<int> maListe (5,0); std::vector<int>	Cree une liste maListe de int avec 5 valeurs de départ valant tous 0 **Pour les fichier.h**
maListe[0] = 118218;	Modifie la première case de la liste par 118218
maListe.push_back(8);	Ajoute une nouvelle case avec la valeur 8 a maListe
maListe.pop_back();	Supprime la **dernière** ligne de la liste seulement
maListe.size()	Renvoie la taille de la liste

Lecture/Ecriture fichier:

ofstream monFlux("scores.txt"); nomFichier.c_str()	Ouvre le fichier score.txt **en ecriture** et stocke dans monflux Pour le cas où le nom du fichier est dans un string de variable nomFichier
ofstream monFlux("scores.txt", ios::app);	Pour l'écriture a partir de la fin du fichier
monFlux << "Bonjour, je suis une phrase écrite dans un fichier." << endl;	Ecrit dans le fichier monFlux
ifstream monFlux("scores.txt");	Ouvre le fichier score.txt **en lecture** et stocke dans monflux
string ligne; getline(monFlux, ligne);	Variable ligne Lit **UNE** ligne du fichier et l'enregistre dans la variable ligne
string mot; monFlux >> mot;	Variable mot Lit **UN** mot du fichier et le stocke dans la variable mot **(converti en int si la variable est de type int)**
char a; monFlux.get(a);	Variable a Lit **UNE** lettre et la stocke dans la variable a
REFAIRE SI ON VEUT LE/LA SUIVANT(E)	

while(getline(fichier, ligne))
{
 	
}
	Lit ligne a ligne jusqu'à la fin du fichier

flux.close();	Ferme le fichier
int position = fichier.tellp(); int position = fichier.tellg();	Récupère la position du curseur dans un fichier Pour mode écriture Pour mode lecture
flux.seekg(nombreCaracteres, position); flux.seekp(nombreCaracteres, position);	Déplace le curseur de nombreCaractere à partir de position Pour mode écriture Pour mode lecture **le début du fichier :** ios::beg **la fin du fichier :** ios::end **la position actuelle :** ios::cur