

**UNIVERSITÉ GASTON BERGER DE SAINT-LOUIS**

----

**U.F.R DE SCIENCES APPLIQUÉES ET DE TECHNOLOGIE**

---

**DÉPARTEMENT D'INFORMATIQUE**

---

**Algorithmique & Programmation 2**

**Travaux Pratiques**

**Licence 1 – MASS/MPI**

---

On ne peut rien apprendre aux gens. On peut seulement les aider à découvrir qu'ils possèdent déjà en eux tout ce qui est à apprendre.



*Citation de célébrité*

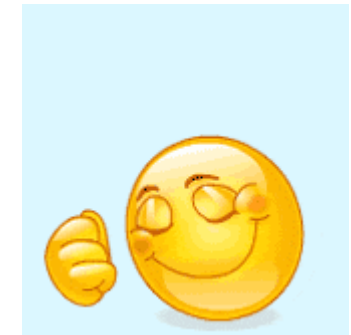
**Galilée**

*Astronome, Mathématicien, Physicien, Scientifique (1564 - 1642)*

Vous êtes ....



ou....



# Quelques algorithmes + mini projet

Quelques algorithmes

# Recherche dichotomique

- Algorithme récursif

Entrée : Tableau ordonné (trié)  $t$ , un intervalle  $[gauche, droite]$  avec  $0 \leq gauche \leq droite < \text{taille}$  et une valeur (élément) à chercher.


- Sortie :

- indice  $pos$  et  $t[pos]$  tel que  $t[pos] = el$ , où  $el$  est une valeur saisie au clavier à recherche dans  $t$  ;
- ou 0 (Faux) si  $el$  ne se trouve pas dans  $t$ .


- Principe de l'algorithme : La recherche s'effectue dans l'intervalle  $[gauche, droite]$  de la manière suivante :

- Initialisation :

11	21	32	45	57	61	72	76	80	82	101
----	----	----	----	----	----	----	----	----	----	-----

  
 $gauche = 0$

  
 $milieu = (gauche + droite) / 2$

  
 $droite = n$

- Valeur à chercher :  $v=72$

- $milieu = 5 : v=72 > t[5] \rightarrow$  Recherche à droite
- $gauche = milieu + 1 = 6$
- $milieu = gauche + 1 = 8$
- $72 < t[8] \rightarrow$  Recherche à gauche
- $droite = milieu - 1 = 7$
- $milieu = droite - 1 = 6$
- $72 = t[6]$
- Fin /\* REVOIR LES CALCULS ET PRENDRE AVEC UN EXEMPLE DE 5 ÉLÉMENTS \*/

# Recherche dichotomique

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#define MAX 50
/* Procédure saisie */
void saisie (int t[], int n)
{ int i;
  for (i=0;i<=n-1;i++)
  { printf("t[%d]=",i);
    scanf("%d",&t[i]); }
}
/* Procédure affichage */
void affiche (int t[], int n)
{ int i;
  for (i=0;i<=n-1;i++)
    printf("%d\t",t[i]);
  printf("\n\n");
}
```

```
int rechDicho (int t[], int n, int gche, int drte, int rech, int *pos, bool *trouve)
{ int milieu;
  do{
    if (drte<gche) {
      *trouve=false;
      return 0; }
    milieu =(gche+drte)/2;
    *pos=milieu;
    if (rech==t[milieu]) {
      *trouve=true;
      return t[milieu]; }
    else {
      if(rech<t[milieu]) {
        drte=milieu-1;
        return rechDicho(t,n,gche,milieu,rech, pos,trouve); }
      else {
        gche=milieu+1;
        return rechDicho(t,n,milieu,drte,rech,pos,trouve); }
    } } while (gche<=drte && !trouve);
}
```

# Recherche dichotomique

```
int main()
{
    int t[MAX], n,gche, drte, el,pos;
    int reponse;
    bool trouve;
    do {
        do {
            printf("ENTREZ LE NOMBRE D'ELEMENTS DU TABLEAU :");
            scanf("%d",&n);
        }while (n<2 || n>50);

        printf("SAISIE DES ELEMENTS DU TABLEAU :\n");
        saisie (t,n);
        printf("AFFICHAGE DES ELEMENTS DU TABLEAU :\n");
        affiche(t,n);
        printf("ENTREZ LA VALEUR A CHERCHER :");
        scanf("%d",&el);
        gche=0; drte=n;
```

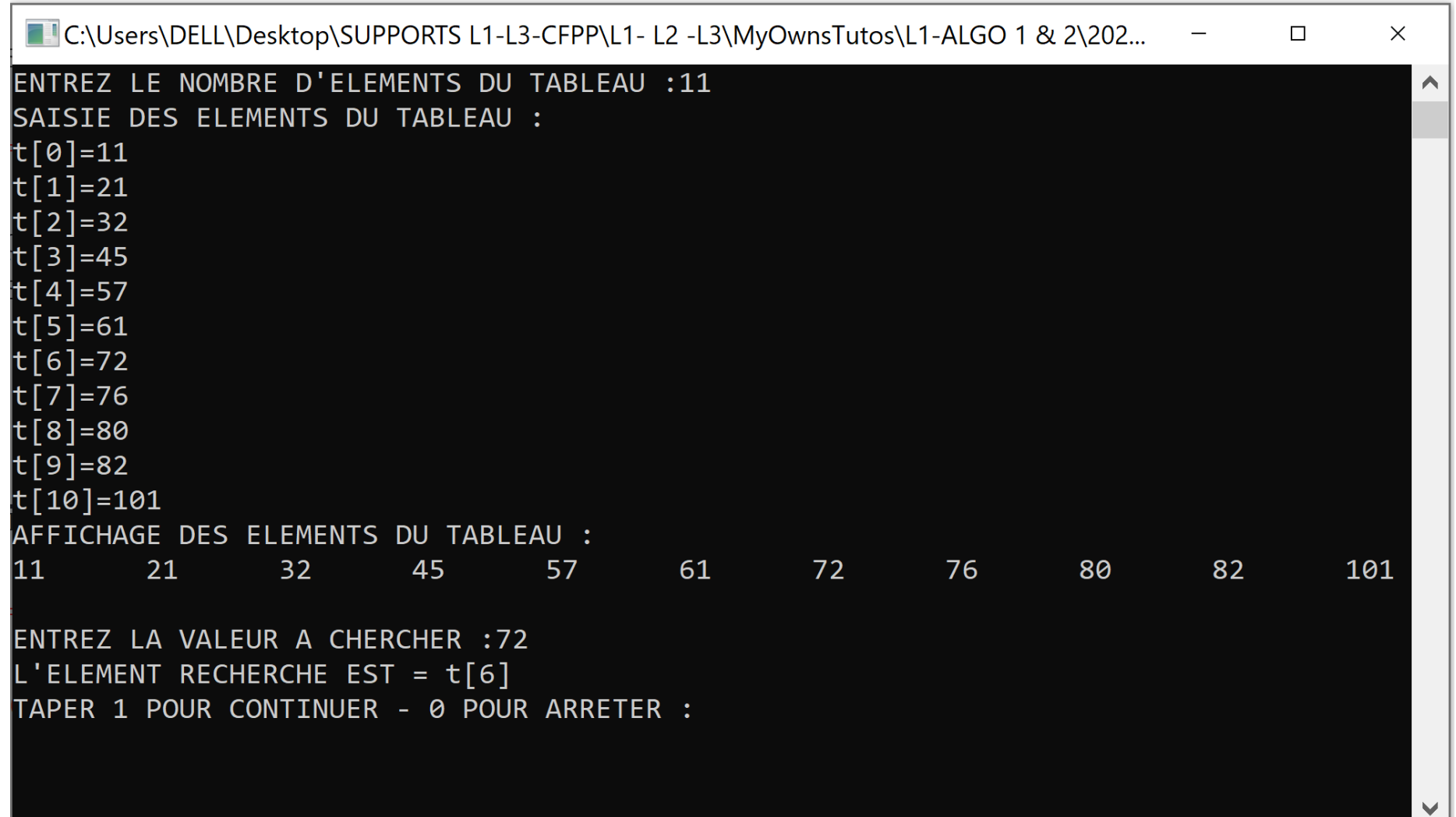
# Recherche dichotomique

```
rechDicho(t,n,gche,drte,el,&pos,&trouve);
if (!trouve)
{
    printf("%d N'EST PAS DANS LE TABLEAU !\n",el);
}
else {
    printf("L'ELEMENT RECHERCHE EST = t[%d]\n",pos); }
do {
    printf("TAPER 1 POUR CONTINUER - 0 POUR ARRETER :");
    scanf("%d", &reponse);
}while (reponse !=1 && reponse !=0);
printf("\n");
} while (reponse !=0);

/* system("PAUSE"); */
printf("\n\n");
printf("RECHERCHE DICH0 : ALGORITHME RECURSIF TERMINE...AU REVOIR !");
return 0;
}
```

# Recherche dichotomique

- Exemple d'exécution :



```
C:\Users\DELL\Desktop\SUPPORTS L1-L3-CFPP\L1- L2 -L3\MyOwnsTutos\L1-ALGO 1 & 2\202...  
ENTREZ LE NOMBRE D'ELEMENTS DU TABLEAU :11  
SAISIE DES ELEMENTS DU TABLEAU :  
t[0]=11  
t[1]=21  
t[2]=32  
t[3]=45  
t[4]=57  
t[5]=61  
t[6]=72  
t[7]=76  
t[8]=80  
t[9]=82  
t[10]=101  
AFFICHAGE DES ELEMENTS DU TABLEAU :  
11      21      32      45      57      61      72      76      80      82      101  
  
ENTREZ LA VALEUR A CHERCHER :72  
L'ELEMENT RECHERCHE EST = t[6]  
TAPER 1 POUR CONTINUER - 0 POUR ARRETER :
```



# Recherche dichotomique - Variante : Algorithme itératif

```
booléen dico(Entier t [], Entier n, Entier v)
booléen trouve = Faux
Entier gche = 0      /* indice du premier élément du tableau t (à gauche) */
Entier drte = n-1    /* indice du dernier élément du tableau t (à droite) */
Entier milieu ;
DÉBUT
    Tant Que (!trouve) ET (gche <= drte) Faire
        Si gche > drte Alors
            trouve = faux                      /* Rien à chercher ! */
        FinSi
        milieu = (gche + drte)/2;              /* indice du pivot  $\geq 0$  */
        Si t[milieu] == v Alors
            trouve = Vrai                     /* le pivot est la valeur cherchée */
        Sinon
            Si v < t[m] Alors
                drte = milieu-1 ;             /* recherche à gauche */
            Sinon
                gche = milieu+1; /* recherche à droite */
        FinSi
    FinTantQue
    retourner trouve /* Vous pouvez aussi renvoyer la position pivot (milieu) pour indiquer l'élément égal à la valeur cherchée */
FIN
```

# Tri rapide – Quick Sort (Charles Antony Richard HOARE -1961)

- **Principe :**

- Basé sur une fonction de partition
- **Entrée :** Un ensemble fini de valeurs  $X$  et une valeur pivot  $p$  ;
- **Sortie :**
  - Ensemble  $X_1$  tel que  $x_i \in X$  et  $x_i < p$
  - Ensemble  $X_2$  tel que  $y_i \in X$  et  $y_i \geq p$ .

- **Démarche :**

- Choisir un élément  $p$  appelé pivot ;
- Placer à gauche les éléments inférieurs à  $p$  ;
- Placer à droite les éléments supérieurs à  $p$  ;
- Trier récursivement la partie de droite et celle de gauche.

# Tri rapide – Quick Sort (Charles Antony Richard HOARE -1961)

- Admettons que les éléments de l'ensemble à partitionner sont mémorisés dans les cellules d'indice compris entre  $i$  et  $j$  avec  $i < j$

$p \leftarrow t[\text{max}]$

$i \leftarrow \text{min}; j \leftarrow \text{max}-1$

**Répéter**

**Tant que**  $t[i] < p$  **faire**  $i \leftarrow i+1$

**Tant que**  $t[j] > p$  **faire**  $j \leftarrow j-1$

**Si**  $i < j$  **Alors**

permuter  $t[i]$  avec  $t[j]$

$i \leftarrow i+1; j \leftarrow j-1$

**Sinon**

permuter  $t[i]$  avec  $t[\text{max}]$

**retourner**  $i$

## Fiche 3 – Exercice 4 :1-2-3

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAXC 32
#define MAXEL 10
typedef char CHAINE32[MAXC];
typedef CHAINE32 TAB10[MAXEL];
void tri(TAB10 *t[], int n)
{   int j, k;
    CHAINE32 *temp;
    for (j = 0; j < n; j++)
        for (k = 0; k < n; k++)
            if (strcmp(t[j], t[k])<0)
            {
                temp=t[j];
                t[j]=t[k];
                t[k]= temp;
            }
}
```

```
int main()
{ int i,nel= 6;
  TAB10 *tpers[]= {"Saliou", "Anna", "Abdou","Modeste",
                  "Zakaria","Solange"};
  printf("LISTE DES PERSONNES A TRIER :\n");
  printf("=====\n");
  for (i=0;i<nel;i++)
    printf("%s\n",tpers[i]);
  printf("\n");
  tri(tpers,nel);
  printf("LISTE DES PERSONNES PAR ORDRE ALPHABETIQUE :\n");
  printf("=====\n");
  for (i=0;i<nel;i++)
    printf("%s\n",tpers[i]);
  printf("\n");
  system("PAUSE");
  return 0;
}
```

# Fiche 3 – Exercice 4

**1-2-3**

```
LISTE DES PERSONNES A TRIER :  
=====
```

Saliou
Anna
Abdou
Modeste
Zakaria
Solange

```
LISTE DES PERSONNES PAR ORDRE ALPHABETIQUE :  
=====
```

Abdou
Anna
Modeste
Saliou
Solange
Zakaria

```
Appuyez sur une touche pour continuer...
```

# Quelques algorithmes + mini projet

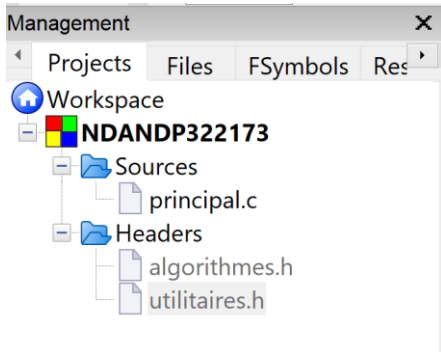
Mini projet

# Mini projet

- Réaliser un projet permettant d'exécuter :
  1. La recherche dichotomique (**algorithme itératif**), l'insertion d'un élément dans un tableau ordonné, la fusion de deux tableaux ordonnés (utiliser une fonction-procédure insertion pour insérer les éléments ; l'un des tableaux sera résultat de la fusion);
  2. Les algorithmes de tris suivants :
    - Sélection ordinaire;
    - Insertion séquentielle ;
    - Tri rapide.
- INDICATIONS :
  - I1 : votre projet comprendra trois fichiers – le programme principal et deux en-têtes - que vous nommerez comme suit :
    - **principal.c** comprenant uniquement la fonction principale main () avec des déclarations de variables locales, des appels de fonctions-procédure et, bien sûr, l'en-tête des directives **#include** nécessaires ;
    - **algorithmes.h** comprenant toutes les fonctions-procédures : **recherche()**, **insertion()**, **fusion()**, **sélection()**, **insertion()**, **triRapide()**;
    - **utilitaires.h** comprenant les fonctions-procédures **menu()** (qui affiche le menu principal), **sousMenu1 ()**, **sousMenu2()**, **saisie ()**, **permutation()**, **affichage ()**.
  - I2 : La définition de type demandée à l'exercice 2 de la fiche 3 sera effectuée dans le fichier en-tête (header) **utilitaires.h**;
  - I3 : Vous spécifierez – bien sûr - les arguments formels de chacune des fonctions-procédures ci-dessus en gardant les mêmes identificateurs de noms (voir I1: ci-dessus) .

# Mini projet

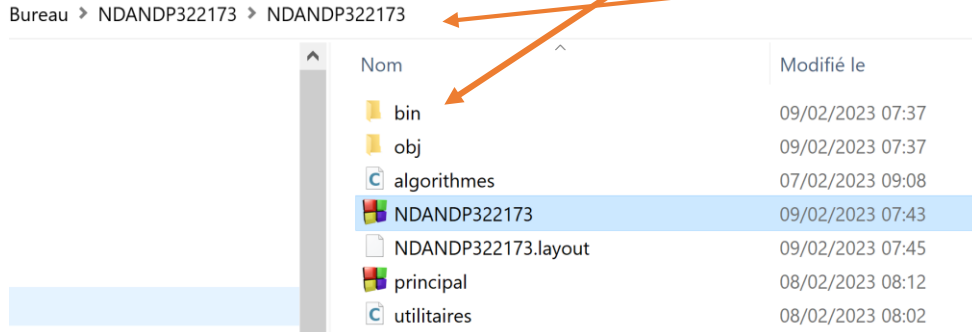
- Aperçu de votre espace de travail (sous Code::Blocks)



**Indications :** Pour obtenir cette perspective :

1. Créez un nouveau dossier – sur le bureau – que vous **renommez avec vos initiales suivies de votre numéro d'étudiant (e)**, par **exemple : NDANDP322173** ;
2. Démarrez Code::Blocks et créez un nouveau projet – **portant le même nom que ci-dessus** – que vous enregistrerez dans le dossier nouvellement créé (et renommé en conséquence) ;
3. Retirer main.c du projet (Sélection – clic droit – Remove file from project);
4. Créez les 3 fichiers sources (menu File – New – File, suivre les étapes) dans cet ordre : utilitaires – algorithmes – principal.

- Rappel : Pour afficher/masquer l'espace de travail : View -> Manager**
- En plus des codes sources à envoyer, vous joindrez un fichier **rapport.pdf** contenant les captures d'écrans suivantes:
  - Espace de travail (voir ci-dessus);
  - Dossiers contenant votre projet et les sous-dossiers, **avec le chemin d'accès au-dessus**, comme illustré ci-dessous :





# Mini projet

- Votre rapport contiendra également :
  - la liste des fichiers contenus dans les sous-dossiers **bin** et **obj** ainsi que leur description;
  - les réponses aux questions suivantes :
    - Qu'est-ce que le **debug** ?
    - Qu'est-ce que la **release** ?
    - Quelle est la différence entre **debug** et **release** ?
    - Comment est obtenu – **à partir de quel fichier** - l'exécutable dans ces deux sous-dossiers ?
  - des captures d'écran d'un exemple d'exécution de votre programme à partir de l'un des sous-dossiers (au choix) contenant le fichier exécutable, **en veillant à bien montrer que l'exécution est réalisée en dehors de Code::Blocks.**

# Mini projet

```
/* FONCTION-PROCÉDURE QUI PERMET L'INSERTION D'UNE VALEUR DANS UN TABLEAU ORDONNÉ */
```

```
void insertX(TAB100 t, int x, int n) /* TAB100 est un type tableau d'entiers de taille maximale 100 à définir dans le fichier utilitaires.h */  
{  
    int pos=0, j;  
    /* LE TABLEAU ÉTANT TRIÉ DANS L'ORDRE CROISSANT,  
    ON LE PARCOURT JUSQU'A TROUVER L'ÉLÉMENT PLUS PETIT QUE x A INSÉRER*/  
    while (pos<n && x>t[pos])  
        pos++; /* ON INCRÉMENTE TANT QUE x EST PLUS GRAND */  
  
    /* DÉCALAGE A DROITE (LE TABLEAU ÉTANT ORDONNÉ) */  
    for (j=n-1; j>=pos-1; j--)  
        t[j+1]=t[j];  
    /* INSERTION DE L'ÉLÉMENT */  
    t[pos] = x;  
}
```

**NB** : A utiliser obligatoirement dans la fonction-procédure de fusion de deux tableaux, **sans utilisation d'un troisième tableau.**

# Mini Projet

- Au démarrage de votre programme, c'est le menu suivant qui devra être affiché :

```
MENU PRINCIPAL
=====
1.Recherche-Insertion-Fusion
2.Algorithmes de tris
3.Fin
=====
VOTRE CHOIX SVP (1/2/3):
```

- Exemple d'exécution :

```
MENU PRINCIPAL
=====
1.Recherche-Insertion-Fusion
2.Algorithmes de tris
3.Fin
=====
VOTRE CHOIX SVP (1/2/3):1
```

```
Recherche-Insertion-Fusion
=====
R.Recherche dichotomique
I.Insertion d'un element
U.Fusion de deux tableaux
F.Fin.
=====
```

```
FAITES VOTRE CHOIX [R(r)/I(i)/U(u)/F(f)] :
```

```
FAITES VOTRE CHOIX [R(r)/I(i)/U(u)/F(f)] :I
ENTREZ LE NOMBRE D'ELEMENTS DU TABLEAU :3
t[0]=21
t[1]=1
t[2]=13
1      13      21
```

```
INSERTION D'UN ELEMENT DANS LE TABLEAU :
```

```
-----
DONNEZ L'ENTIER A INSERER :8
1      8      13      21
```

```
MENU PRINCIPAL
=====
1.Recherche-Insertion-Fusion
2.Algorithmes de tris
3.Fin
=====
VOTRE CHOIX SVP (1/2/3):
```

```
MENU PRINCIPAL
=====
1.Recherche-Insertion-Fusion
2.Algorithmes de tris
3.Fin
=====
VOTRE CHOIX SVP (1/2/3):2
```

```
Algorithmes de tris
=====
S.Tri par selection
I.Insertion sequentielle
R.Tri rapide
F.Fin.
=====
```

```
VOTRE CHOIX SVP(S(s)/I(i)/R(r)/F(f)) :S
ENTREZ LE NOMBRE D'ELEMENTS DU TABLEAU :
```

# Mini Projet

- Au démarrage de votre programme, c'est le menu suivant qui devra être affiché :

```
MENU PRINCIPAL
=====
1.Recherche-Insertion-Fusion
2.Algorithmes de tris
3.Fin
=====
VOTRE CHOIX SVP (1/2/3):
```

- Exemple d'exécution(suite) :

```
MENU PRINCIPAL
=====
1.Recherche-Insertion-Fusion
2.Algorithmes de tris
3.Fin
=====
VOTRE CHOIX SVP (1/2/3):2

Algorithmes de tris
=====
S.Tri par selection
I.Insertion sequentielle
R.Tri rapide
F.Fin.
=====

VOTRE CHOIX SVP(S(s)/I(i)/R(r)/F(f)) :S
ENTREZ LE NOMBRE D'ELEMENTS DU TABLEAU :
```

```
ENTREZ LE NOMBRE D'ELEMENTS DU TABLEAU :4
t[0]=24
t[1]=8
t[2]=3
t[3]=12
3      8      12      24
```

```
MENU PRINCIPAL
=====
1.Recherche-Insertion-Fusion
2.Algorithmes de tris
3.Fin
=====
VOTRE CHOIX SVP (1/2/3):
```

# Mini Projet

- Au démarrage de votre programme, c'est le menu suivant qui devra être affiché :

```
MENU PRINCIPAL
=====
1.Recherche-Insertion-Fusion
2.Algorithmes de tris
3.Fin
=====
VOTRE CHOIX SVP (1/2/3):
```

- Exemple d'exécution(suite) :

```
MENU PRINCIPAL
=====
1.Recherche-Insertion-Fusion
2.Algorithmes de tris
3.Fin
=====
VOTRE CHOIX SVP (1/2/3):3

AU REVOIR ...
```

- **NB:** Le dernier message AU REVOIR (avec un temps d'attente) n'est pas imposé mais constituerait un bonus si réalisé.

- Envoyer votre travail à : [oumar.sy@ugb.edu.sn](mailto:oumar.sy@ugb.edu.sn)

- **NB** :

- **Objet de votre mel** : Projet ALGO 2
- **Corps de votre mel** :

Bonjour Monsieur,

Ci-joint, notre travail cité en objet.

Prénom (s) et nom (s).

**PJ** :

- Codes sources (principal.c, utilitaires.h, algorithmes.h)
- Rapport (rapport.pdf)