# Texte contenu dans un fichier "text.txt"

**Ut vulputate mi a odio porta  a vestibulum sem mollis  Fusce vulputate lacus faucibus purus sagittis  finibus sagittis velit lobortis  Curabitur pretium dapibus mi  id condimentum mauris gravida vitae  Vivamus nec massa et leo posuere blandit vel vitae elit  Morbi ac massa ultrices  varius nunc nec pharetra nunc  Phasellus malesuada nunc et sapien condimentum fringilla  Nullam lobortis et est ac auctor  Suspendisse quis felis dictum risus varius placerat  Nunc ornare  sem in ornare consequat  nisi felis lacinia arcu  vitae cursus lorem augue quis sem  Nunc nisi massa  dapibus eu dolor sit amet finibus efficitur nunc  Sed efficitur non nisi elementum auctor  Mauris sollicitudin purus ut ante placerat hendrerit  Proin neque diam  vestibulum eu bibendum id  feugiat sit amet ex  Mauris consequat blandit ex  vitae gravida nunc rutrum sed  Quisque ut pellentesque mauris  Nunc porttitor felis lacinia laoreet suscipit  Donec consequat  nunc sed viverra eleifend  ante ante lobortis lacus  nec malesuada diam est vel ligula  Suspendisse nec dolor arcu Aenean venenatis ultrices justo vel porta  Sed sit amet tellus id risus hendrerit varius  Fusce hendrerit mollis lacinia  Nunc lobortis vulputate euismod Duis quis dolor tortor  Etiam vel augue quis augue tempus lacinia ac ut nisi  Donec condimentum turpis et ipsum suscipit.**

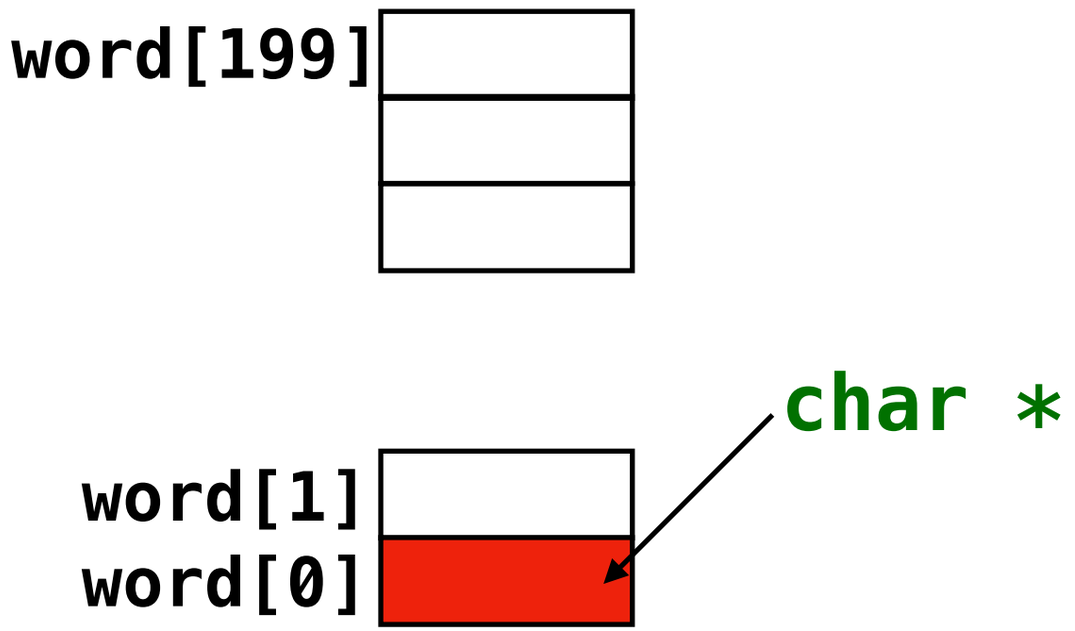
# Objectifs du TD20200326 :

1. accéder au fichier texte, calculer le nombre de caractères,
2. allouer une zone mémoire permettant d'accueillir le texte,
3. transférer le texte depuis le fichier vers la zone mémoire,
4. créer une fonction "wc" qui reçoit le texte et qui compte et retourne le nombre de mots (séparateurs admis : espace, virgule ou point),
5. **créer une fonction "ws" qui reçoit le texte et qui va placer chaque mot dans un tableau "word",**
6. **agrémenter la fonction "ws" pour éviter les doublons dans le tableau "word",**
7. **agrémenter la fonction "ws" en ajoutant un tableau qui contient le nombre d'occurrences pour chaque mot.**

# Ut vulputate mi a odio porta  a vestibulum

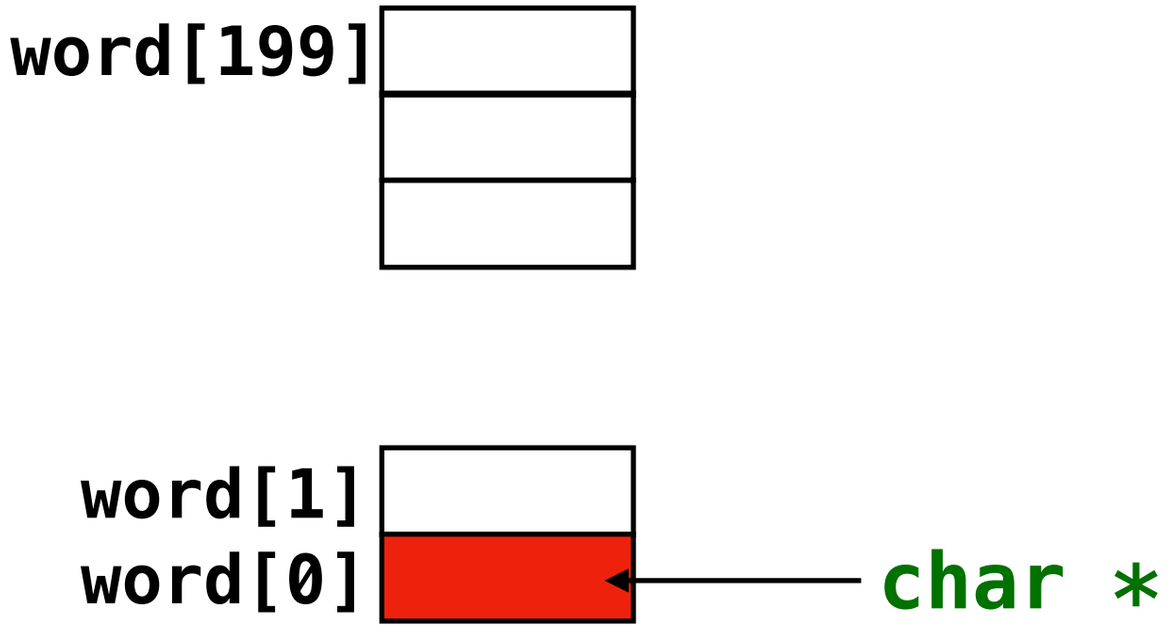
Tableau word[]

word[0]="Ut"
word[1]="vulputate"
word[2]="mi"
.
.
.
word[199]="suscipit"




```
char * word[200]; // tableau à taille fixe de mots
char **word=NULL; // tableau dynamique de mots
```

```
double t[30];   // tableau à taille fixe de double
double *t=NULL; // t=(double*)malloc(30*sizeof(double));
```




```
length= (stop-start+1)+1;
word[0]=(char*)malloc(length*sizeof(char));
```

s="Ut vulputate mi a odio porta  a vestibulum…"

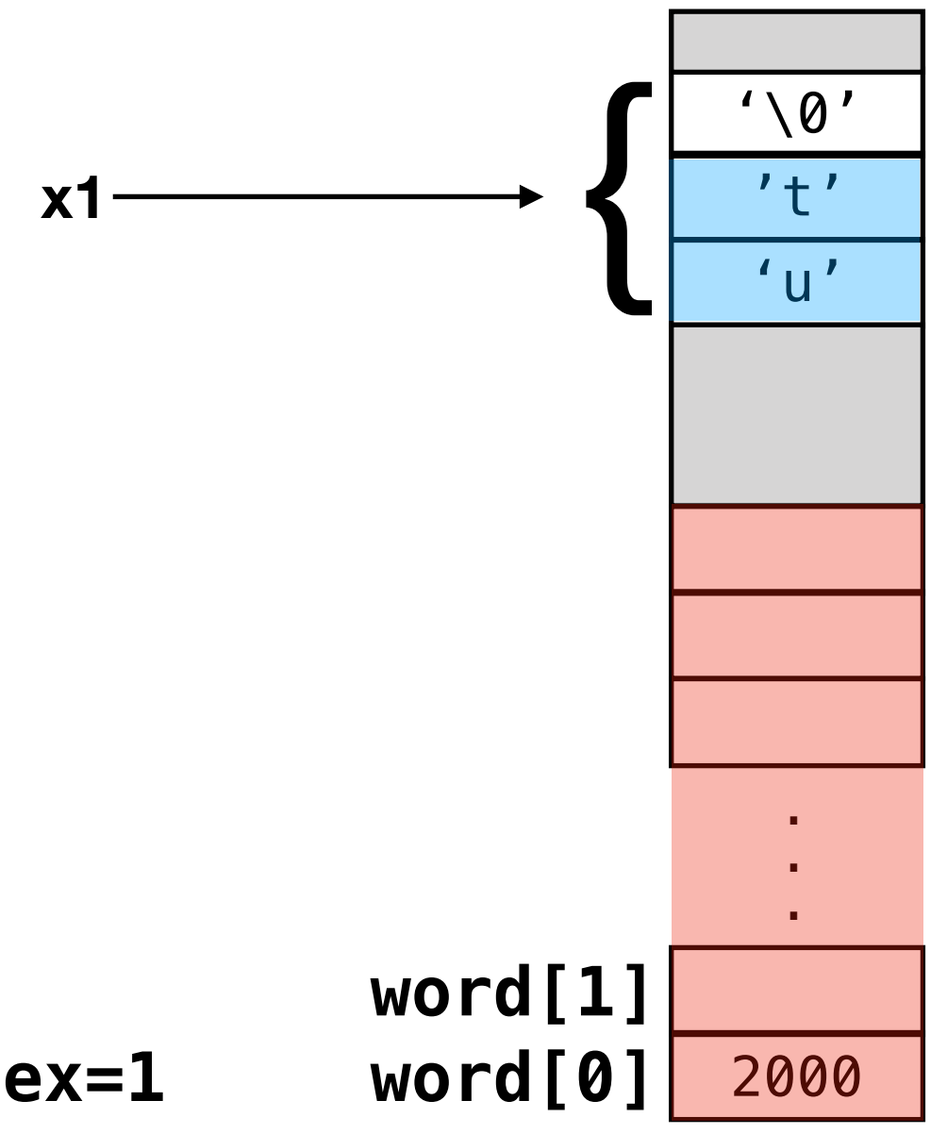
start

stop

x2

‘\0’

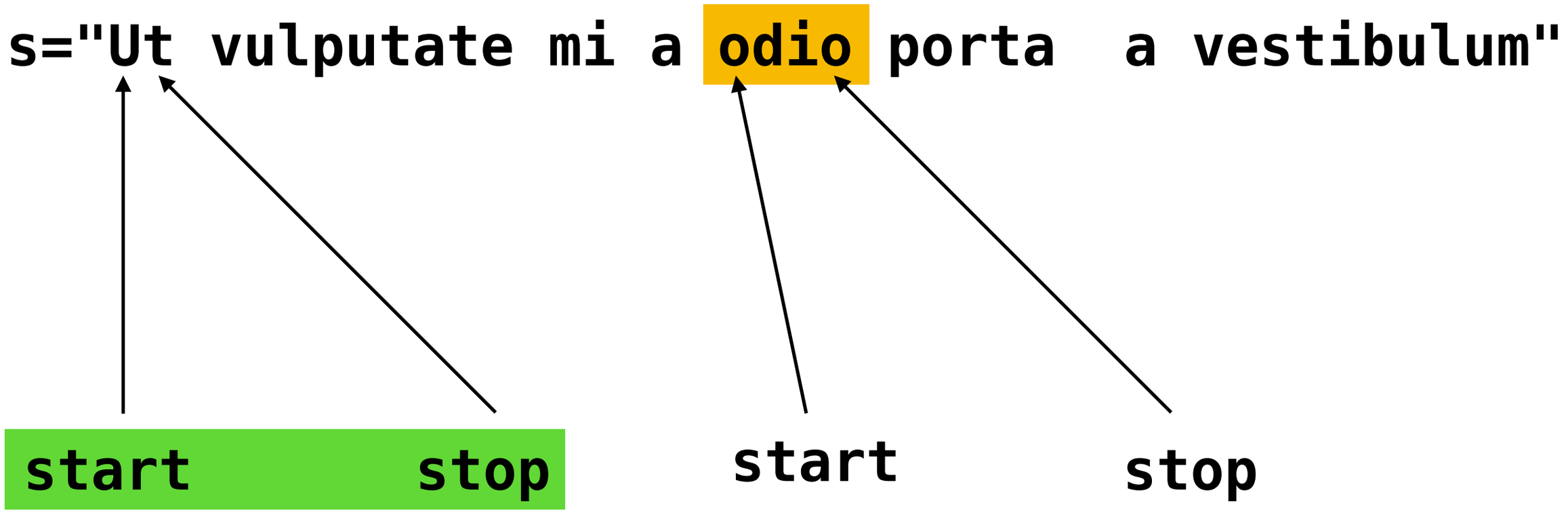
```
x1=word[0];
*(stop+1)=‘\0’;
x2=start;
if(0==strcmp( x1, x2)){
 // mot existe déjà
}
else {
}
*(stop+1)=‘ ’;
```




wordIndex=1

```
char *s="Un chaton";

nb caractères utiles : 9
nb caractères pour la chaîne : 9+1 (‘\0)


fileSize=9;
```

```
// lire 8 octets depuis f et en garder 1 pour le ‘\0’
fgets(text, fileSize, f);


// lire 9 octets depuis f et en garder 1 pour le ‘\0’
fgets(text, 1+fileSize, f);
```

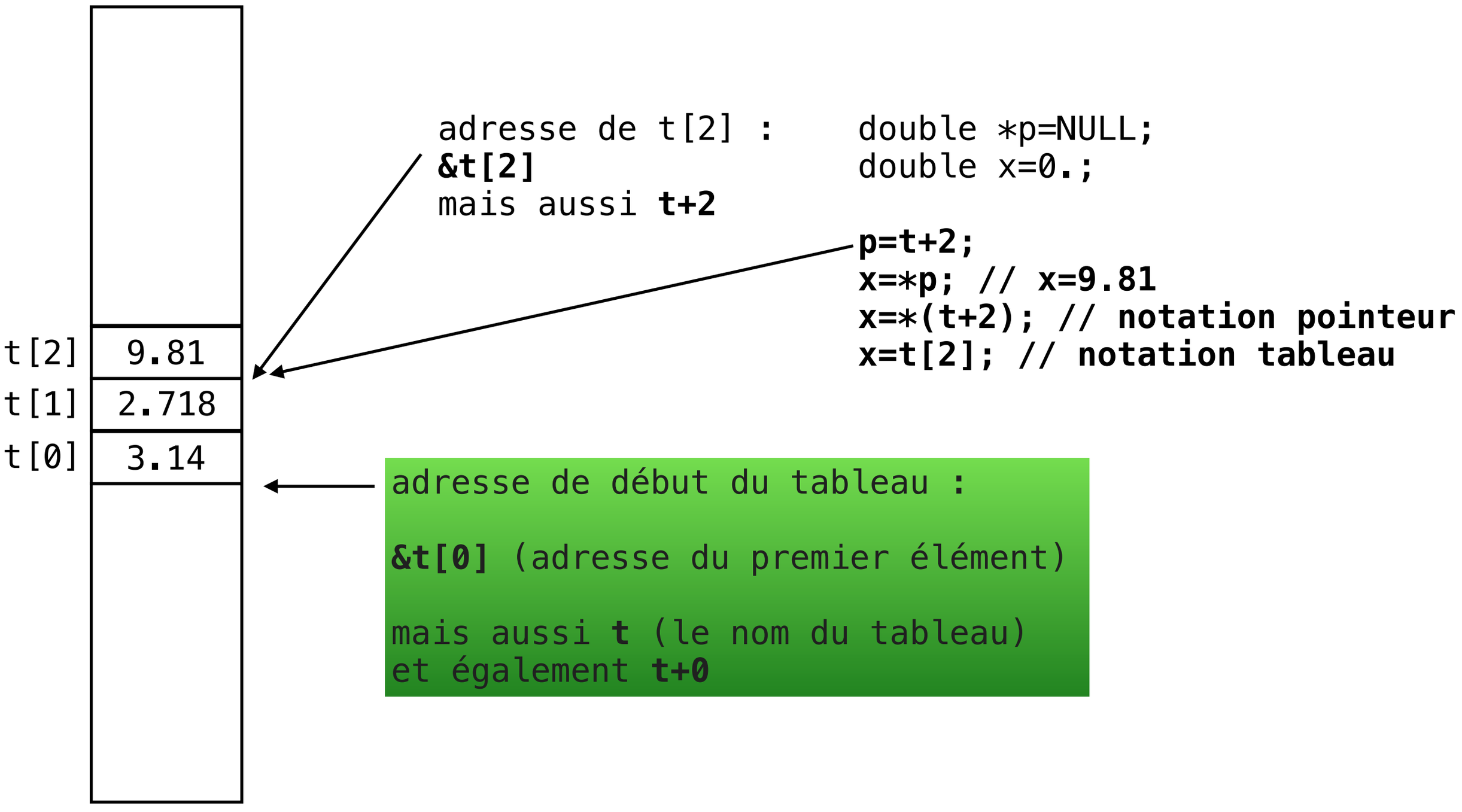



boucle pour afficher les lettres de start à stop

```
Ut
vulputate
mi
a
odio
porta
…
```

```
char *start=NULL;
char *stop=NULL;
char *p=NULL;

for(p=start;p<=stop;p++) {
   putchar(*p);
}
puts("");
```

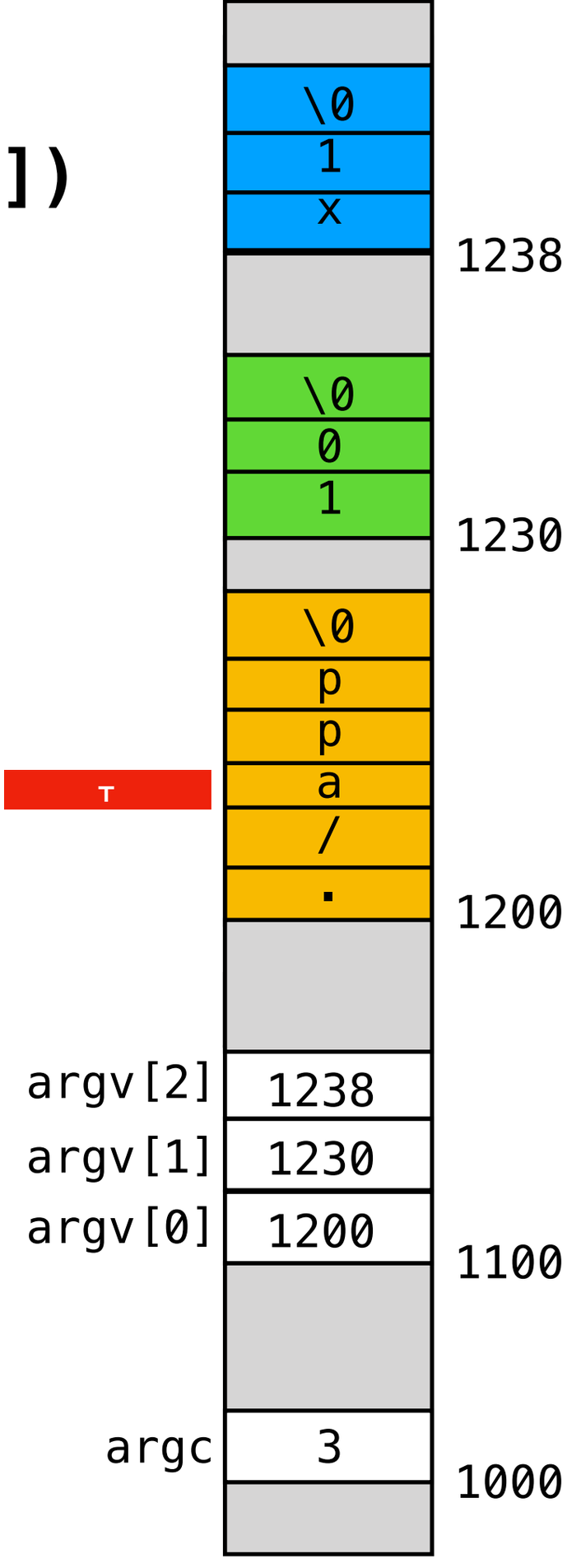
```
#define N 3

double t[N]={ 3.14, 2.718, 9.81 };
```



```
int main(int argc, char* argv[])
```

```
./app 10 x1
```

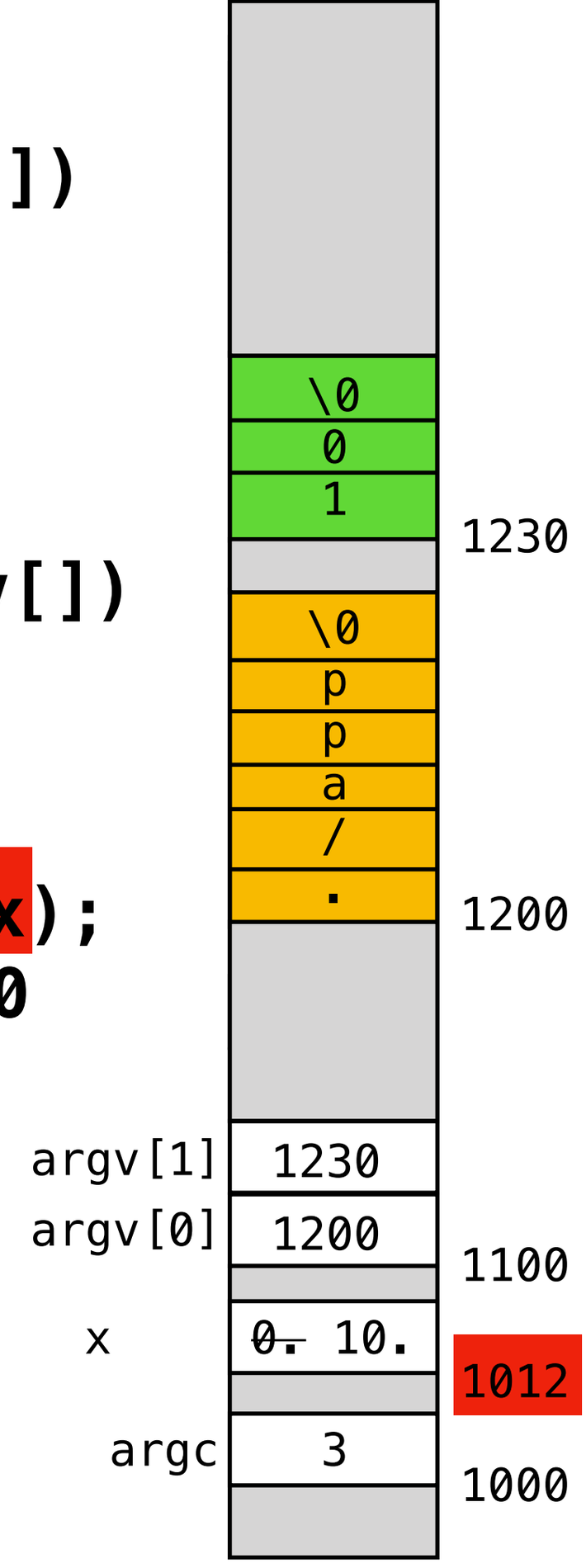
```
argv[0][2]='T';
```



```
int main(int argc, char* argv[])
```

```
./app 10
```

```
int  main(int  argc,  char*  argv[])
{
   double x=0.;
    x=atof(argv[1]);
    ret=sscanf(argv[1],"%lf", &x);
   printf("%lf",x); // 10.000000
}
```



s="Ut vulputate mi a odio porta  a vestibulum…"

start stop
length=2

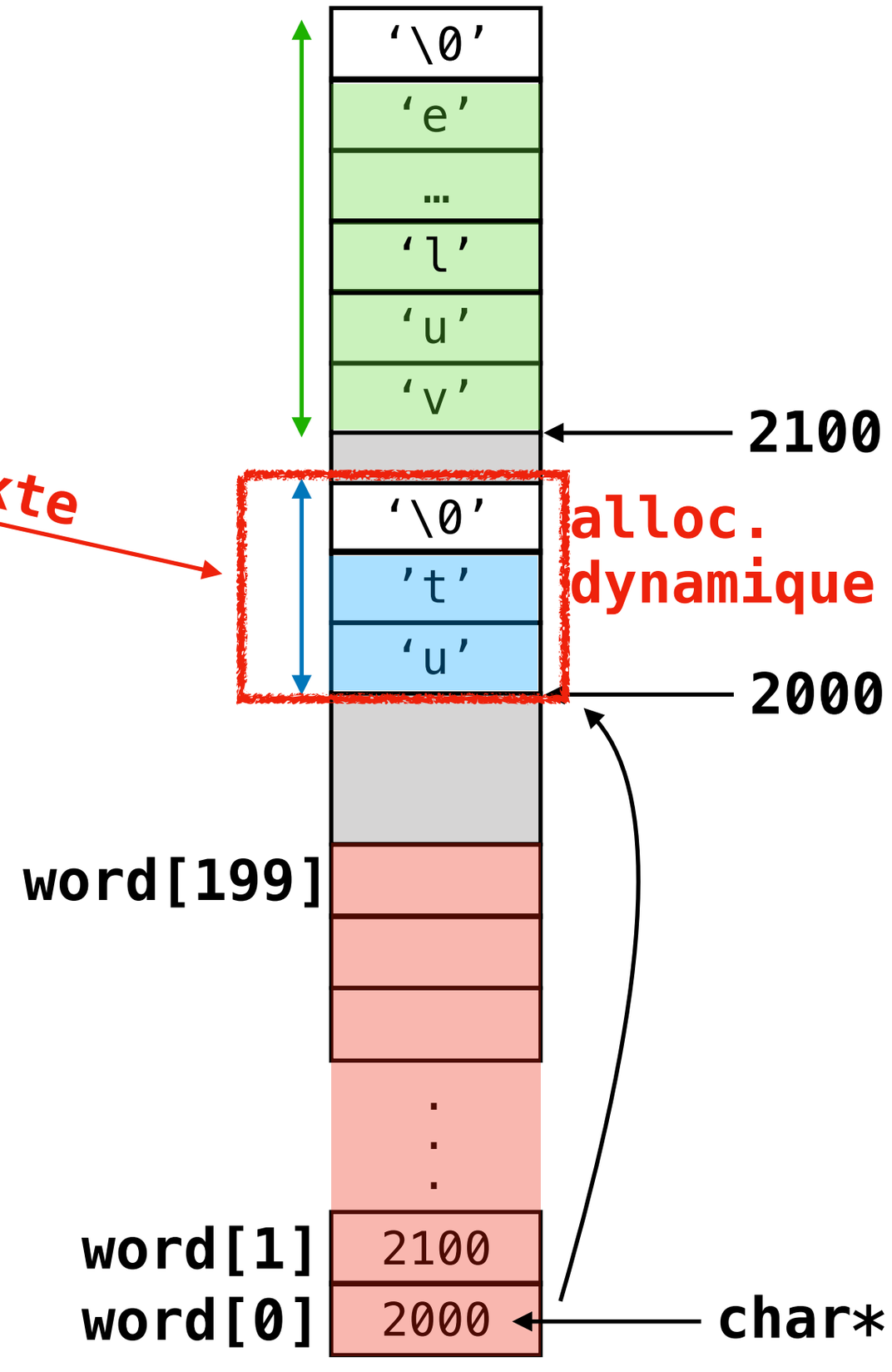
start stop
length=9

copie du texte

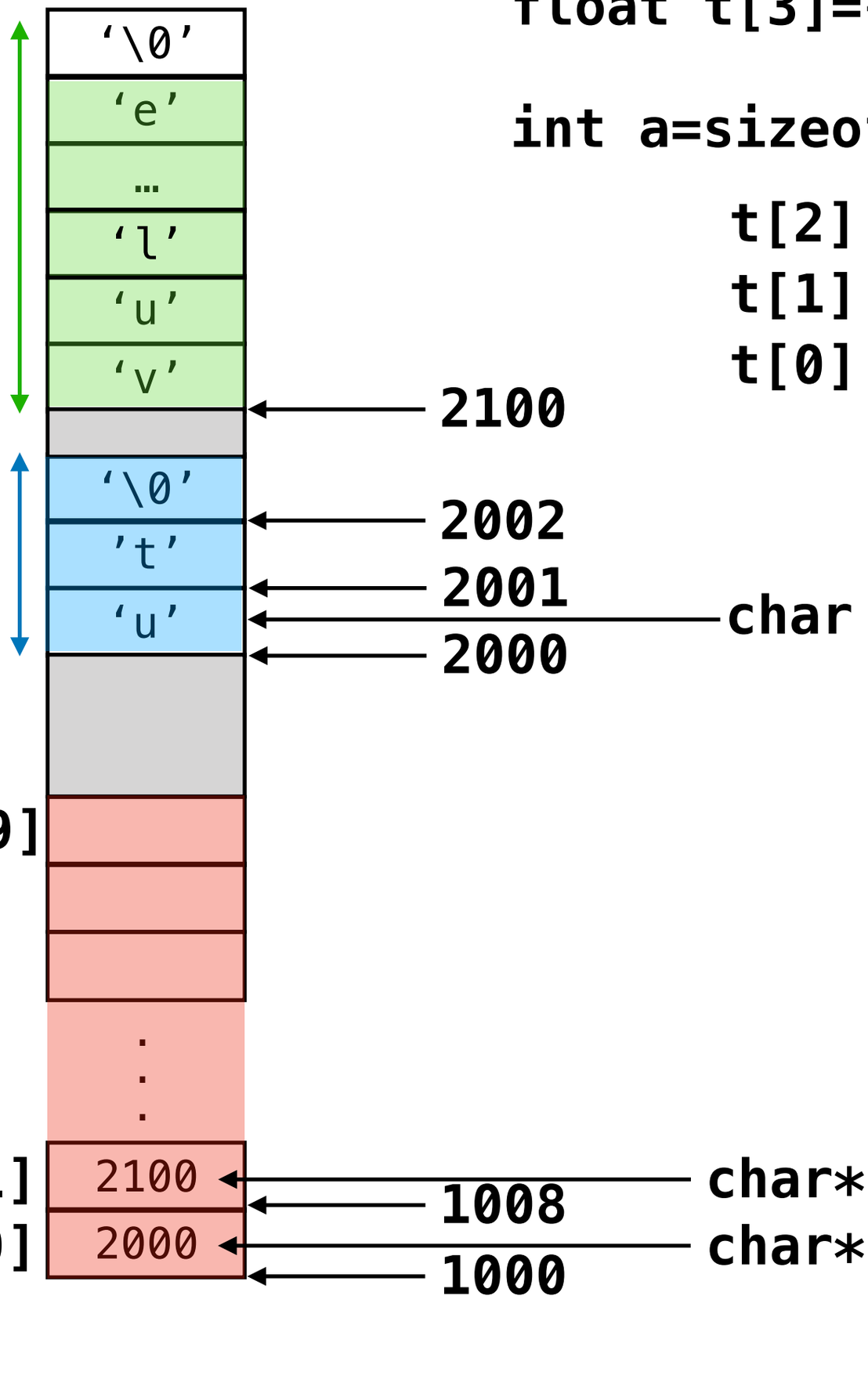
```
// tableau dyn. de double
double *t=NULL;
```

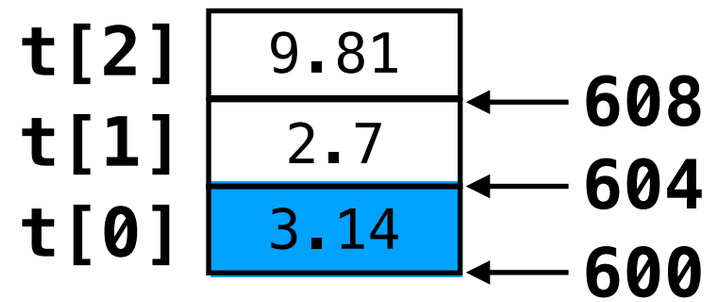
```
// tableau dyn. de chaine
char* *word=NULL;
```

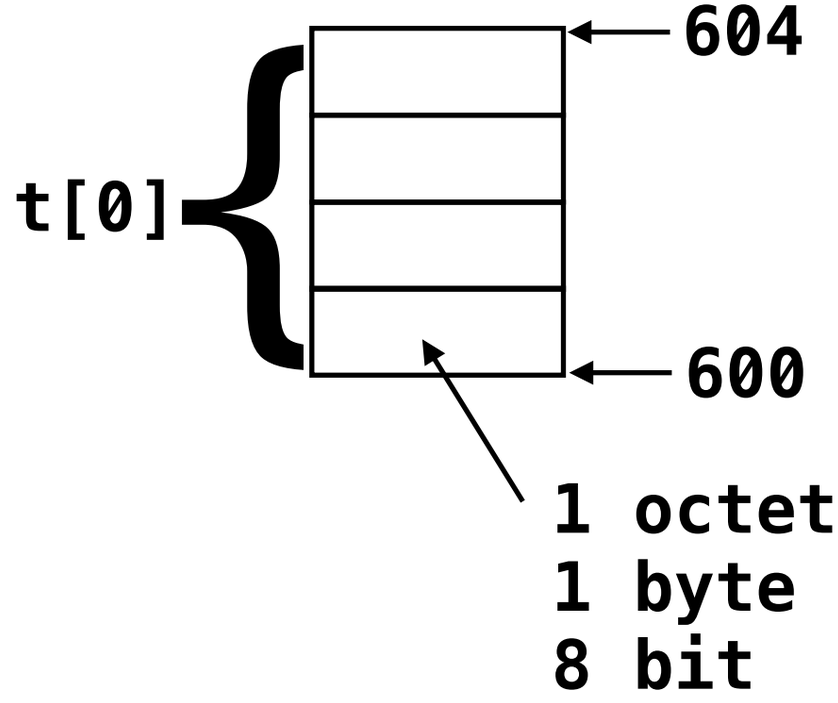
word pointe sur une chaine



```
float t[3]={3.14,2.7,9.81};

int a=sizeof(float); // a=4 octets
```



```
if(!inWord && !((s[index]==‘ ‘)||s[index]==‘.’))) {
}
```

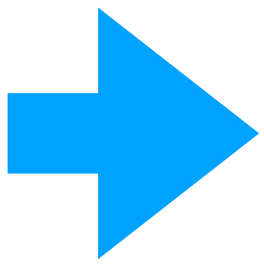
```
 if(!inWord && ((s[index]!=‘ ‘)||s[index]!=‘.’))) {
}
```

**S=! (A || B)**
**S=!A && !B**

| A | B | A \|\| B | !(A\|\|B) | !A | !B | !A && !B |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 |

```
s="cheval chat cheval chien."
```

```
word[3] chien
word[2] cheval
word[1] chat
word[0] cheval
```



```
word[2] chien
word[1] chat
word[0] cheval
```

```
s="cheval chat cheval chien."
```

length=5

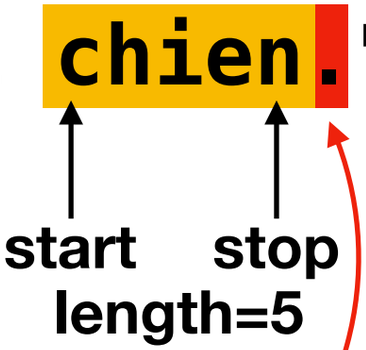
start

stop

```
strcmp("cheval","chien") => !=0

strcmp("chien","chien") => ==0
```

indexWord=2

s="cheval chat cheval chien."




word[1] "chat"
word[0] "cheval"

```
boucle: 2 conditions de sortie
1.   on a rien trouvé                bool wordFound=false
2.   on a trouvé le même mot          bool finished=false

index=0;
*(stop+1)='\0';
while(!finished) {
  if(strcmp(start, word[index])==0) {
    finished=true;
    wordFound=true;
  }
  index++;
  if(index==indexWord) {
    finished=true;
  }
}
*(stop+1)=' ';
if(wordFound) => il ne faut pas insérer le mot dans
                 le tableau word.
```