Listes avec vecteurs

9 septembre 2024 15:45

1) Revoir devoir ▸ bien supprimer, nullptr
2) Classe Tableau en exercice
   └ importance du copieur
3) Autres concepts C++ ⇒ livre (lvalue, smart pointers, const pointers, ...)

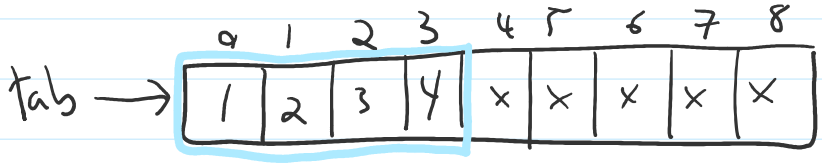
TDA : liste

élément

- ajouter un élt en début / milieu / fin
- supprimer un élt en début / milieu / fin
- obtenir / modifier l'élt en position i ✓ ✓
- obtenir la taille

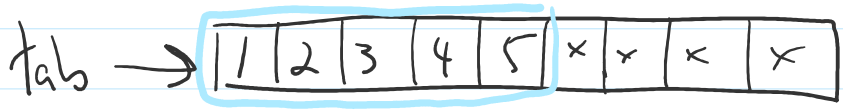
Implémentation 1 : vector

- Tableau avec espace pour :
  - objets stockés — nbelem
  - objets futurs — cap



nbelem = 4
cap = 9

push_back(5)

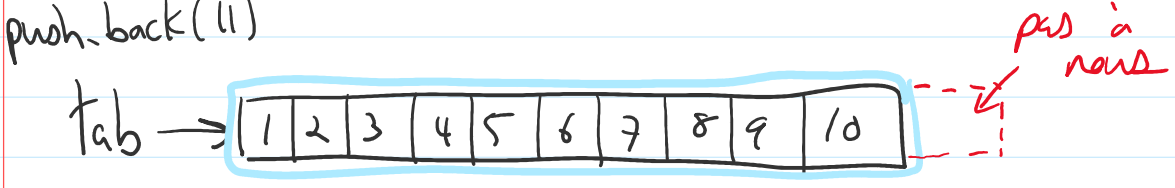


nbelem = 5
cap = 9

...

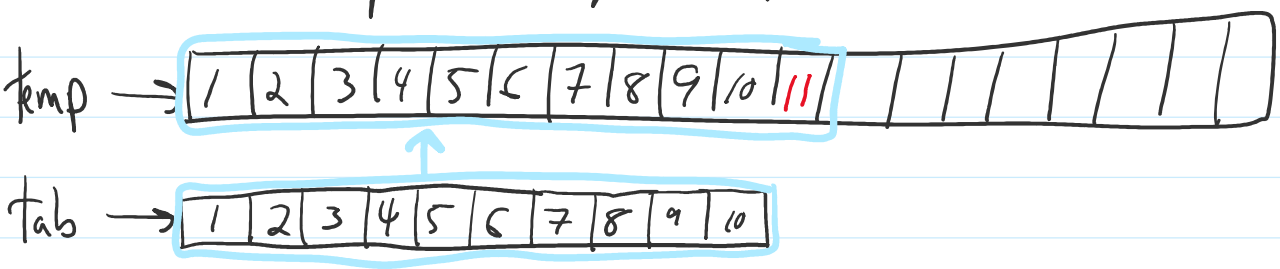
Que faire si nbelem == cap?

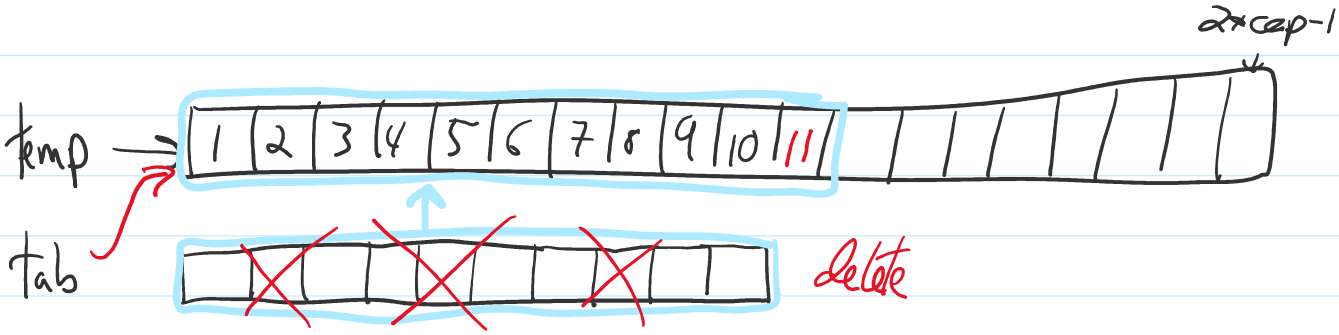
push_back(11)



On double la taille de notre tableau!

- créer un tableau temp de taille 2*cap
- copier tab[0] à tab[nbelem] dans temp
- supprimer tab
- tab = temp cap = 2*cap





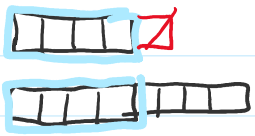
- fonction reserve(size_t newcap) réservera l'espace requis

```
template <typename TYPE>
class vector {
private:
    TYPE* tab;
    size_t nbelem;
```

```
    TYPE* tab;
    size_t nbelem;
    size_t cap;
public:
    //...
};
```

```
vector(){
    nbelem = 0;
    cap = 0;
    tab=nullptr;          // pour reserve
    reserve(10);          // pré-déclare cap=10
}
```

```
void push_back(const TYPE& val){
    if (nbelem >= cap){
        reserve( 2*cap);
    }
    tab[nbelem] = val;    //copie val
}
```



```
void reserve(size_t newcap){
    TYPE* temp = new TYPE[newcap];
    for(size_t i=0; i<nbelem; ++i)
        temp[i] = tab[i];
    if (tab != nullptr)
        cleanup();
```

```
        if (tab != nullptr)
            delete [] tab;

        cap = newcap;
        tab = temp;
    }

    TYPE& operator[](size_t i){
        return tab[i];
    }

    TYPE& at(size_t i){
        if (i >= nbelem)
            throw "argh";
        return tab[i];
    }

    size_t size(){
        return nbelems;
    }

    ~vector(){
        if (tab)
            delete [] tab;
    }

    void cleanup(){
        if (tab){
            delete [] tabs;
            tab=nullptr;
        }
```

cleanup();

cleanup();

```
    }
}
```
tab=nullptr;

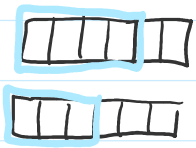
//copieurs!

```
vector& operator=(const vector& src){
        clean();
        nbelem = src.nbelem;
        reserve(src.cap);

        for(size_t i=0; i<nbelem; ++i){
            tab[i]=src[i];
        }
        return *this;
}
vector(const vector& src){
    (*this) = src;
}

void pop_back(){    //enlève dernier elt
    if (nbelem>0)
        nbelem--;
}

void resize(size_t size){
    if(size<=nbelem){
```



```
    if (size <= nbelem){
        nbelem = size;
    }
    else{
        reserve(size);
    }
}
```

- Ajoutons une fct find(TYPE& val)
  - retourne première pos qui contient val
  - -1 si non-présent
  - pas la façon idéale, voir itérateurs
  - pas dans la Standard Library (SL)

```
int find(const TYPE& val){

    for(size_t i=0; i<nbelem; ++i){
        if (tab[i] == val)
            return i;
    }
    return -1;
}
```