Support de cours

Cours:

Initiation à la programmation (en C++)

Vidéo:

Fonctions spécifiques des vector (partie 2)

Concepts (extraits des sous-titres générés automatiquement) :

Tableau dynamique d'entiers. Fonctions spécifiques. Dernier élément du tableau. Fonction saisie. Valeurs. Tableau dynamique. Deuxième convention. Partie valeur de retour. Type de la taille. Différentes étapes. Nombre négatif. Deuxième argument. Conventions suivantes. Exemples typiques d'appels. Tab d'entiers.

vers la recherche de séquences vidéo
(dans Initiation à la programmation (en C++).)

vers la vidéo

EPFL

Tableaux : fonctions spécifiques `vector`

(Partie 2)

Initiation à la programmation (C++)

Vincent Lepetit, Jean-Cédric Chappelier et Jamila Sam

...

notes

résumé

0m 0s

Exemple

Ecrivons une fonction qui (ré)initialise un tableau dynamique d'entiers en les demandant à l'utilisateur, qui peut

- ajouter des nombres strictement positifs au tableau
- recommencer au début en entrant *0*
- effacer le dernier élément en entrant un nombre négatif

```
Saisie de 3 valeurs :
Entrez la valeur 0 : 5
Entrez la valeur 1 : 2
Entrez la valeur 2 : 0
Entrez la valeur 0 : 7
Entrez la valeur 1 : 2
Entrez la valeur 2 : -4
Entrez la valeur 1 : 4
Entrez la valeur 2 : 12

-> 7 4 12
```

Prenons un exemple qui utilise plusieurs de ces fonctions spécifiques.

notes

résumé

0m 1s

Exemple

Ecrivons une fonction qui (ré)initialise un tableau dynamique d'entiers en les demandant à l'utilisateur, qui peut

- ajouter des nombres strictement positifs au tableau
- recommencer au début en entrant *0*
- effacer le dernier élément en entrant un nombre négatif

```
Saisie de 3 valeurs :
Entrez la valeur 0 : 5
Entrez la valeur 1 : 2
Entrez la valeur 2 : 0
Entrez la valeur 0 : 7
Entrez la valeur 1 : 2
Entrez la valeur 2 : -4
Entrez la valeur 1 : 4
Entrez la valeur 2 : 12

-> 7 4 12
```

L'exemple en question ça va être d'écrire une fonction qui va initialiser ou réinitialiser un tableau dynamique d'entiers en demandant à l'utilisateur de rentrer des valeurs suivant les conventions suivantes. Si l'utilisateur entre des nombres strictement positifs alors on va les ajouter au tableau. Si l'utilisateur entre 0 alors on va tout annuler et on va recommencer au début. Si l'utilisateur enfin entre un nombre négatif, alors ça veut juste dire qu'on va effacer le dernier élément du tableau. Par exemple, supposons que l'on demande à l'utilisateur d'entrer trois valeurs et que l'utilisateur donc au départ, entre la valeur 5. Il va donc créer ici le tableau dynamique avec un seul élément : 5. Puis, l'utilisateur rentre la valeur 2, donc on rajoute un élément qui a 2. Puis l'utilisateur a tapé 0,

notes

résumé

0m 5s

Exemple

Ecrivons une fonction qui (ré)initialise un tableau dynamique d'entiers en les demandant à l'utilisateur, qui peut

- ajouter des nombres strictement positifs au tableau
- recommencer au début en entrant *0*
- effacer le dernier élément en entrant un nombre négatif

```
void saisie(vector<int>& vect, size_t taille = 4)
{
  vect.clear();
  cout << "Saisie de " << taille << " valeurs :" << endl;
  while (vect.size() < taille) {
    cout << "Entrez la valeur " << vect.size() << " : ";
    int val;
    cin >> val;
    if ((val < 0) and (not vect.empty())) { vect.pop_back();     }
    else if (val == 0)                    { vect.clear();        }
    else if (val > 0)                     { vect.push_back(val); }
  }
}
```

ce qui correspond à la convention de dire : on recommence tout à zéro et donc on repart du tableau vide. Ensuite l'utilisateur tape 7, donc on recrée un tableau dynamique à un seul élément qui est 7 puis il retape 2 puis il tape -4. Ça c'est la deuxième convention, ça veut dire que à ce moment-là on efface juste le dernier élément du tableau. Puis l'utilisateur tape 4, puis il tape 12. On a donc ici atteint les trois valeurs que l'on voulait entrer, le programme s'arrête et le tableau contient bien 7, 4 et 12. On cherche à écrire une fonction. Si vous vous souvenez des différentes étapes pour écrire une fonction, on vient ici de spécifier clairement ce que l'on veut faire : le "quoi". La deuxième étape maintenant c'est de s'intéresser au prototype donc regardons des exemples typiques d'appels. Supposons qu'on ait donc un tableau dynamique tab d'entiers, on pourrait par exemple, notre fonction on va vouloir décider qu'elle s'appelle saisie, c'est son nom et on voudrait par exemple saisir en indiquant tab et en indiquant la taille, par exemple 5 éléments. On pourrait aussi décider que arbitrairement, si on ne donne pas la taille, alors à ce moment-là ça veut dire que on aura à saisir 4 éléments, par exemple. Un autre exemple d'utilisation serait ici d'avoir un tableau dynamique qui est déjà initialisé. La différence ici c'était que tab était un tableau dynamique vide alors qu'ici tab2 est un tableau dynamique qui contient 12 éléments et à ce moment-là on pourrait par exemple dire qu'on va saisir dans tab2 les 12 éléments, donc tab2.size(), qu'il contient. Donc on va les ressaisir en plaçant comme deuxième argument, ici, sa taille. Voilà donc différents exemples d'utilisation de notre fonction saisie. Reste maintenant donc à l'écrire. Pour écrire la fonction saisie, on commence par se demander quel doit être son prototype.

notes

résumé

1m 1s

Exemple

Ecrivons une fonction qui (ré)initialise un tableau dynamique d'entiers en les demandant à l'utilisateur, qui peut

- ajouter des nombres strictement positifs au tableau
- recommencer au début en entrant *0*
- effacer le dernier élément en entrant un nombre négatif

```
vector<int> tab;

saisie(tab, 5); // saisie de 5 éléments

saisie(tab); // saisie de 4 éléments

vector<int> tab2(12);

saisie(tab2, tab2.size());
```

Tout d'abord, ses paramètres. On a vu pour cela, dans l'appel,

notes

résumé

3m 1s

Exemple

Ecrivons une fonction qui (ré)initialise un tableau dynamique d'entiers en les demandant à l'utilisateur, qui peut

- ajouter des nombres strictement positifs au tableau
- recommencer au début en entrant *0*
- effacer le dernier élément en entrant un nombre négatif

z = saisie(tab); ?

```
void saisie(vector<int>& vect, size_t taille = 4)
{
  vect.clear();
  cout << "Saisie de " << taille << " valeurs :" << endl;
  while (vect.size() < taille) {
    cout << "Entrez la valeur " << vect.size() << " : ";
    int val;
    cin >> val;
    if ((val < 0) and (not vect.empty())) { vect.pop_back();     }
    else if (val == 0)                    { vect.clear();        }
    else if (val > 0)                     { vect.push_back(val); }
  }
}
```

que l'on allait devoir passer à un tableau comme premier argument et qu'on allait avoir un deuxième argument optionnel qui serait une taille. Donc ça nous donne les deux paramètres suivants : un tableau et une taille. Quel est le type de la taille ? Pour cela, on respecte la norme, qui nous dit que les tailles de tableaux sont de type size_t. Donc on déclare taille de type size_t. Est-ce que taille peut prendre une valeur par défaut ? Oui, on a vu dans les appels qu'on pouvait avoir une valeur par défaut, c'est-à-dire ne pas spécifier la taille et on va arbitrairement décider donc que cette taille par défaut, est 4. Passons au premier paramètre. Le premier paramètre c'est le tableau dynamique vect. Est-ce que ce tableau doit être modifié par la fonction ? Oui, c'est bien le rôle de la fonction saisie que de saisir les valeurs du tableau, donc comme le tableau est modifié, on va le passer par référence. Son type, c'est un type tableau dynamique d'entiers, donc passé par référence. Ce qui nous donne donc la partie paramètres de la fonction. Pour la partie valeur de retour, je vous rappelle, on se pose la question : est-ce que ça a un sens d'écrire z=saisie(tab) ? Est-ce que ça aurait un sens ? Qu'est-ce que serait la valeur de z de cette saisie d'un tableau ? Personnellement je ne vois pas quel sens ça pourrait avoir. Et donc le type de retour ici sera void pour dire que la fonction ne retourne rien. On a donc terminé avec le prototype et maintenant on peut s'intéresser au comment s'intéresser au corps de la fonction. Donc pour cela, ce que l'on va commencer par faire c'est de partir du vecteur vide. Donc on va commencer par vider le vecteur vect ici que l'on a reçu comme argument, c'est le même nom que j'utilise, et on va le

notes

résumé

3m 6s

Exemple

Ecrivons une fonction qui (ré)initialise un tableau dynamique d'entiers en les demandant à l'utilisateur, qui peut

- ajouter des nombres strictement positifs au tableau
- recommencer au début en entrant *0*
- effacer le dernier élément en entrant un nombre négatif

```
void saisie(vector<int>& vect, size_t taille = 4)
{
  vect.clear();
  cout << "Saisie de " << taille << " valeurs :" << endl;
  while (vect.size() < taille) {
    cout << "Entrez la valeur " << vect.size() << " : ";
    int val;
    cin >> val;
    if ((val < 0) and (not vect.empty())) { vect.pop_back();     }
    else if (val == 0)                    { vect.clear();        }
    else if (val > 0)                     { vect.push_back(val); }
  }
}
```

vider avec la fonction spécifique clear. Ensuite, on va afficher à l'utilisateur saisie de le nombre d'éléments que l'on veut saisir.

notes

résumé

Exemple

Ecrivons une fonction qui (ré)initialise un tableau dynamique d'entiers en les demandant à l'utilisateur, qui peut

- ajouter des nombres strictement positifs au tableau
- recommencer au début en entrant *0*
- effacer le dernier élément en entrant un nombre négatif

```
void saisie(vector<int>& vect, size_t taille = 4)
{
  vect.clear();
  cout << "Saisie de " << taille << " valeurs :" << endl;
  while (vect.size() < taille) {
    cout << "Entrez la valeur " << vect.size() << " : ";
    int val;
    cin >> val;
    if ((val < 0) and (not vect.empty())) { vect.pop_back(); }
    else if (val == 0)                    { vect.clear();    }
    else if (val > 0)                     { vect.push_back(val); }
  }
}
```

saisie de donc par exemple 3 valeurs puis ensuite, on va faire une boucle tant que l'on n'a pas saisi toutes les valeurs que l'on veut. Donc tant que la taille actuelle du tableau vect que l'on est en train de manipuler, est strictement inférieure à la taille demandée. Alors à ce moment-là tant que cette condition est vraie, on va afficher le message "entrez la valeur" Mais quelle valeur ? A un moment donné supposons que le tableau est partiellement saisi. Par exemple on a déjà saisi les valeurs 7 et 2, et on voudrait donc demander la troisième valeur ici qui va venir être ajoutée. Quelle est la position de cette valeur qui n'est pas encore dedans ? La position du dernier élément c'est "la taille du tableau - 1" et donc cette position ici à venir, c'est bien la position, c'est bien la taille du tableau. Donc ça s'écrit vect.size(). Et donc on va ici afficher vect.size pour dire entrez la valeur, numéro de la valeur qui va être saisie. Ensuite donc on prévoit de lire une valeur lue au clavier la valeur entrée par utilisateur. Donc on lit sur ces lignes, la valeur en question et l'on va maintenant donc traiter les trois cas que l'on a choisi par convention. Donc on avait dit que si la valeur était négative, alors à ce moment-là il fallait que l'on supprime le dernier élément. Donc pour ça il faut garantir que on a bien un dernier élément. Donc si la valeur est négative et que le tableau n'est pas vide. Donc ici, vous voyez, on utilise la fonction spécifique empty qui va nous dire oui ou non, le tableau est vide. Et supprimer le dernier élément suppose que le tableau n'est pas vide. On écrit donc le tableau n'est pas vide : not vect.empty(). A ce moment-là donc, si les des deux conditions la valeur saisie est

notes

résumé

5m 1s

Exemple

Ecrivons une fonction qui (ré)initialise un tableau dynamique d'entiers en les demandant à l'utilisateur, qui peut

- ajouter des nombres strictement positifs au tableau
- recommencer au début en entrant *0*
- effacer le dernier élément en entrant un nombre négatif

```
void saisie(vector<int>& vect, size_t taille = 4)
{
  vect.clear();
  cout << "Saisie de " << taille << " valeurs :" << endl;
  while (vect.size() < taille) {
    cout << "Entrez la valeur " << vect.size() << " : ";
    int val;
    cin >> val;
    if ((val < 0) and (not vect.empty())) { vect.pop_back(); }
    else if (val == 0)                    { vect.clear();    }
    else if (val > 0)                     { vect.push_back(val); }
  }
}
```

négative et le tableau n'est pas vide, alors ce que l'on fait c'est que l'on supprime, avec la fonction spécifique pop_back,

notes

résumé

Exemple

Ecrivons une fonction qui (ré)initialise un tableau dynamique d'entiers en les demandant à l'utilisateur, qui peut

- ajouter des nombres strictement positifs au tableau
- recommencer au début en entrant *0*
- effacer le dernier élément en entrant un nombre négatif

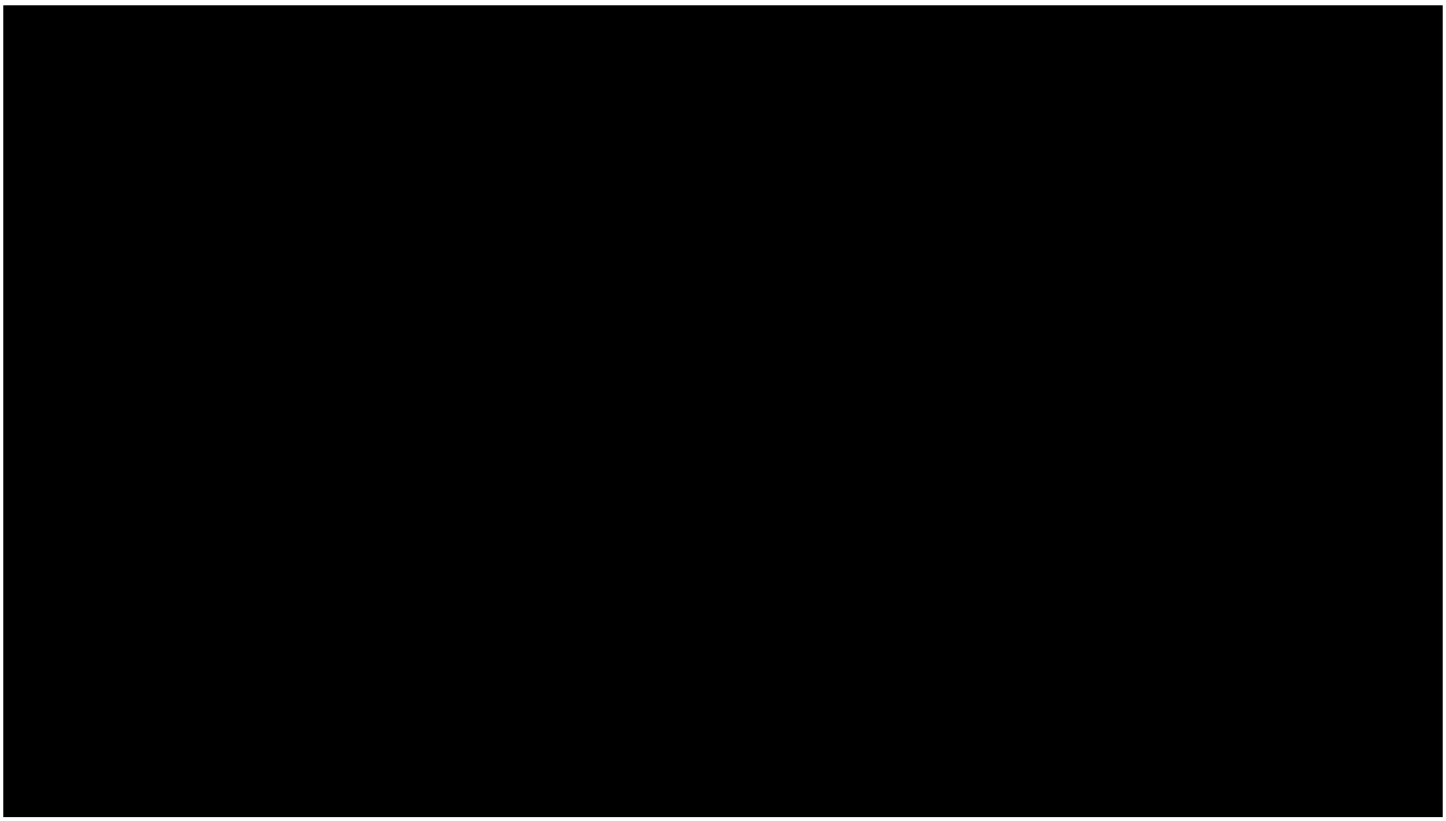
```
void saisie(vector<int>& vect, size_t taille = 4)
{
  vect.clear();
  cout << "Saisie de " << taille << " valeurs :" << endl;
  while (vect.size() < taille) {
    cout << "Entrez la valeur " << vect.size() << " : ";
    int val;
    cin >> val;
    if ((val < 0) and (not vect.empty())) { vect.pop_back();     }
    else if (val == 0)                    { vect.clear();        }
    else if (val > 0)                     { vect.push_back(val); }
  }
}
```

supprime le dernier élément du tableau vect. Sinon, si la valeur saisie est nulle, à ce moment-là la convention c'était de détruire tout le tableau et donc ça se fait à l'aide de la fonction spécifique clear. Donc ici vect.clear().

notes

résumé

7m 1s



Sinon, c'est-à-dire si, soit la valeur est strictement positive, soit elle est strictement négative, et le tableau est vide. Alors pour éliminer ce dernier cas et ne garder que le cas de valeur strictement positive, je fais le test est-ce que val est strictement positif ? Et dans ce cas, je rajoute la valeur avec un push_back, push_back de cette valeur qui va donc bien ajouter la valeur comme dernier élément au tableau dynamique. au tableau dynamique.

notes

résumé

7m 21s