3.2.12(1) Vecteurs et pointeurs : premier essai

Objectifs

- déclarer des variables de type pointeur,
- assigner des valeurs aux variables pointées,
- utiliser l'incrémentation pour déplacer le pointeur dans les emplacements de mémoire adjacents.

Scenario

Regardez le code présenté ci-dessous. Ça n'a pas l'air effrayant, non ?
Votre tâche ne semble pas effrayante non plus - il vous suffit de trouver le plus petit élément du vecteur. Mais il y a une condition - vous ne devez pas utiliser l'indexation [ ]. En d'autres termes, l'utilisation de crochets dans votre code est strictement interdite.

Astuce : Vous pouvez utiliser autant de pointeurs que vous le souhaitez.
Pardonnez que nous ne vous montrons pas de sortie d'échantillon - ce n'est pas une clé. Nous souhaitons que nous puissions voir votre code - c'est beaucoup plus intéressant.

```
using namespace std;

int main(void) {

 int vector[] = { 3, -5, 7, 10, -4, 14, 5, 2, -13 };
 int n = sizeof(vector) / sizeof(vector[0]);

 // Insert your code here

 return 0;
}
```

3.2.12(2) Matrices et pointeurs - un pas à l'intérieur

Objectifs

- comprendre la représentation de la matrice interne,
- utiliser des pointeurs pour accéder à l'emplacement de mémoire désiré,
- combiner les boucles for pour manipuler des tableaux bidimensionnels /

Scenario

Pardonnez-nous de demander - avez-vous terminé l'exercice précédent avec succès ? Si ce n'est pas le cas, veuillez y revenir et réessayer. Croyez-nous - vous ne serez pas capable de résoudre ce problème si vous n'avez pas résolu l'ancien. Désolé, c'est la vie difficile d'un codeur.
Jetez un œil sur le code. Il ne fait presque rien - il sort simplement un contenu de matrice de taille 10 x 10. Comme la matrice est initialement remplie de zéros (pourquoi ?), La sortie ne semble pas très attrayante. C'est pourquoi nous avons besoin de vous ici - votre tâche consiste à remplir la matrice de valeurs qui la transformeront en une table de multiplication. Facile ? Trop facile pour être vrai. Nous allons hausser la barre comme nous l'avons fait auparavant. Vous ne devez pas utiliser de crochets. Vous devez utiliser l'indexation. Ergo - vous devez utiliser des pointeurs.
OK, vous pouvez utiliser des crochets - une fois. Et seulement en déclaration. Nulle part ailleurs. C'est possible. Vraiment.
Quoi qu'il en soit, c'est exactement ce que nous voulons voir sur votre écran :

```
  1   2   3   4   5   6   7   8   9  10
  2   4   6   8  10  12  14  16  18  20
  3   6   9  12  15  18  21  24  27  30
  4   8  12  16  20  24  28  32  36  40
  5  10  15  20  25  30  35  40  45  50
  6  12  18  24  30  36  42  48  54  60
  7  14  21  28  35  42  49  56  63  70
  8  16  24  32  40  48  56  64  72  80
  9  18  27  36  45  54  63  72  81  90
 10  20  30  40  50  60  70  80  90 100
```

Et voici l'extrait que vous devriez développer.

```
#include <iostream>

using namespace std;

int main(void) {

 int matrix[10][10] = { };

 // Insert your code here

 for(int i = 0; i < 10; i++) {
  for(int j = 0; j < 10; j++) {
   cout.width(4);
   cout << matrix[i][j];
  }
  cout << endl;
 }
 return 0;
}
```

3.4.7(1) Anciens problèmes - nouvelles méthodes : fonctions

Objectifs

- concept de définition et d'invocation d'une fonction,
- passer des valeurs d'argument dans une fonction,
- reconstruction du code existant pour répondre aux nouvelles exigences.

Scenario

Il y a quelque temps, nous vous avons demandé de rédiger un programme qui déterminait si une certaine année était un commune ou bissextile. Nous voulons revenir à cette question, mais sous une forme complètement différente.

Nous voulons que vous écriviez une fonction équipée des fonctionnalités suivantes :

- son nom devrait être "isLeap"
- elle accepte un argument de type int représentant le numéro de l'année
- elle renvoie une valeur booléenne : true si l'année est bissextile et false sinon
- elle devrait être muettte ! elle ne doit rien écrire à la sortie - la seule façon dont elle révèle ses actions est la valeur qu'elle renvoie

Nous avons préparé un squelette du programme - remplissez le corps de la fonction avec un contenu approprié !

Nous avons également joint la sortie attendue de votre programme.

```
using namespace std;

bool isLeap(int year) {

 // Insert your code here

}

int main(void) {

 for(int yr = 1995; yr < 2017; yr++)
  cout << yr << " -> " << isLeap(yr) << endl;
 return 0;
}
```

Example output

```
1995 -> 0
1996 -> 1
1997 -> 0
1998 -> 0
1999 -> 0
2000 -> 1
2001 -> 0
2002 -> 0
2003 -> 0
2004 -> 1
2005 -> 0
2006 -> 0
2007 -> 0
2008 -> 1
2009 -> 0
2010 -> 0
2011 -> 0
2012 -> 1
2013 -> 0
2014 -> 0
2015 -> 0
2016 -> 1
```

3.4.7(2) Un pas de plus : trouver la durée des mois

Objectifs

- construire un ensemble de fonctions de coopération,
- signaler les arguments erronés en utilisant une valeur de retour spécifique.

Scenario

Continuons les réflexions de notre codage du temps. Maintenant, vous avez une fonction fiable diagnostiquant la nature d'une année, vous pouvez l'utiliser pour mettre en œuvre une autre fonction importante retournant la longueur de n'importe quel mois (mesurée en jours, bien sûr).

Ecrire une fonction équipée des fonctionnalités suivantes :

- son nom est "monthLength"
- elle accepte deux arguments de type int : numéro de l'année (premier) et numéro du mois (second)
- elle renvoie une valeur int qui représente la longueur du mois spécifié dans une année spécifiée (évidemment, l'année est importante uniquement lorsque month == 2) ou 0 si l'un des arguments d'entrée n'est pas valide
- elle devrait être muette

Nous avons préparé un squelette du programme - remplissez le corps de la fonction avec un contenu approprié !

Nous avons également joint la sortie attendue de votre programme.

Astuce: il existe au moins deux façons d'implémenter la fonction : vous pouvez utiliser switch ou (ce qui semble un peu plus intelligent) déclarer un vecteur stockant des mois - choisissez le style le plus pratique.

```
#include <iostream>

using namespace std;

bool isLeap(int year) {

 // The code you've inserted already
}

int monthLength(int year, int month) {

 // Insert a new code here
}

int main(void) {

 for(int yr = 2000; yr < 2002; yr++) {
  for(int mo = 1; mo <= 12; mo++)
   cout << monthLength(yr,mo) << " ";
  cout << endl;
 }
 return 0;
}
```

Example output

```
31 29 31 30 31 30 31 31 30 31 30 31
31 28 31 30 31 30 31 31 30 31 30 31
```

3.4.7(3) Deuxième étape plus loin : trouver le jour de l'année

Objectifs

- construire un ensemble de fonctions de coopération,
- utiliser des types structurés.

Scenario

Ajoutons un autre outil puissant à notre boîte à outils de calendrier.
Maintenant, écrivez une fonction qui :
• est nommé "dayOfYear"
• elle accepte un argument de type structuré Date - c'est une structure similaire à celle que vous utilisiez avant
• elle renvoie une valeur int correspondant au nombre de jours dans l'année (en supposant que le 1er janvier est le premier jour de chaque année)
• elle devrait être muette
Un squelette de code vous attend ainsi que l'exemple d'entrée et de sortie.

```
#include <iostream>

using namespace std;

struct Date {
 int year;
 int month;
 int day;
};

bool isLeap(int year) {
 // The code you've inserted already
}

int monthLength(int year, int month) {
 // The code you've inserted already
}

int dayOfYear(Date date) {

 // Insert your code here

}

int main(void) {

 Date d;
 cout << "Enter year month day: ";
 cin >> d.year >> d.month >> d.day;
 cout << dayOfYear(d) << endl;
 return 0;
}
```

Example input

```
2016 3 1
```

Example output

```
61
```

Example input

```
2015 3 1
```

Example output

```
60
```

Example input

```
2016 12 30
```

Example output

```
365
```

Example input

```
2015 12 31
```

Example output

```
365
```

3.4.7(4) Troisième étape supplémentaire - compter les jours

Objectifs

- construire un ensemble de fonctions de coopération.

Scenario

Vous êtes maintenant prêt à relever le prochain défi plus ambitieux.
Nous avons besoin d'une fonction qui :

• s'appelle "daysBetween"
• accepte deux arguments de type Date - le premier représente la date «depuis», tandis que le second - la date «jusqu'à»
• renvoie une valeur int étant un nombre de jours écoulés entre les deux dates
• si la date «jusqu'à» est antérieure à la date «depuis», la fonction doit renvoyer -1
• elle devrait être muette

Comme d'habitude, nous fournissons un squelette de code et quelques données de test. Vous fournissez le code et faites les tests - c'est le deal.

```
#include <iostream>

using namespace std;

struct Date {
 int year;
 int month;
 int day;
};

bool isLeap(int year) {
 // The code you've inserted already
}

int monthLength(int year, int month) {
 // The code you've inserted already
}

int dayOfYear(Date date) {
 // The code you've inserted already
}

int daysBetween(Date d1, Date d2) {

 // Insert you code here

}

int main(void) {

 Date since,to;
 cout << "Enter first date (y m d): ";
 cin >> since.year >> since.month >> since.day;
 cout << "Enter second date (y m d): ";
 cin >> till.year >> till.month >> till.day;
 cout << daysBetween(since,till) << endl;
 return 0;
}
```

Example input

```
1901 1 1
2016 1 1
```

Example output

```
42003
```

Example input

```
2001 12 30
2016 12 31
```

Example output

```
5480
```

Example input

```
1999 1 31
1999 12 1
```

Example output

```
304
```

Example input

```
1999 1 2
1999 1 11
```

Example output

```
9
```

Example input

```
1999 2 2
1999 1 11
```

Example output

```
-1
```

3.4.7(5) Un avant-goût de la programmation système - obtention de la date actuelle

Objectifs

- faire usage de services système standard,
- comprendre les exemplaires de l'interface de la fonction système.

Scenario

Nous voulons vous dire un secret : nous allions vous demander d'écrire un programme qui serait capable de compter tous les jours que vous avez vécus depuis votre anniversaire jusqu'à demain. Cela devrait être facile - en fait, vous avez déjà tous les composants nécessaires. Ce serait comme construire une maison en utilisant des blocs fabriqués en usine.

Mais il y a un inconvénient : le programme peut vouloir demander à l'utilisateur la date actuelle. Ce serait un non-sens - un ordinateur demandant à l'homme des données qu'il connaît parfaitement lui-même. Nous avons donc modifié nos plans et maintenant nous voulons que vous écriviez une fonction retournant une structure de type défini précédemment mais remplie avec la date actuelle (au moment de l'invocation de la fonction). Bien sûr, la validité de cette variable est limitée. Vous devriez l'utiliser le plus rapidement possible et ne pas le stocker longtemps car les dates changent. Ne geler pas le temps dans votre code - demandez à votre ordinateur une date à chaque fois que vous en avez besoin.

Voici un ensemble de nos exigences pour la fonction :

- son nom est "Today" (comme ça)
- elle n'utilise aucun argument du tout
- elle renvoie une structure de type Date renseignée avec la date actuelle
- elle devrait être muette

Vous pouvez être un peu perplexe maintenant - ne vous inquiétez pas. Nous allons vous montrer comment gérer le calendrier de l'ordinateur. Ce n'est pas une science de fusée.

Jetez un œil sur le code ci-dessous. Il utilise une technique ancienne et traditionnelle basée sur un ensemble de fonctions classiques dérivées des systèmes POSIX. Un peu moche, mais efficace et portable.

```
#include <iostream>
#include <ctime>

using namespace std;

int main(void) {
 time_t t      = time(NULL);
 tm     tl     = *localtime(&t);

 cout << tl.tm_year+1900 << "-" << tl.tm_mon+1 << "-" << tl.tm_mday << endl;
 return 0;
}
```

Plusieurs questions doivent être clarifiées :

- noter l'inclusion du fichier d'en-tête "ctime": il contient des prototypes pour les fonctions fonctionnant sur des dates et des heures; nous allons utiliser deux d'entre eux
- type "time_t": utilisé pour représenter l'heure d'une manière très étrange : c'est un nombre de secondes écoulées depuis 0:00 AM le 1er janvier 1970. Etrange, n'est-ce pas ?
- invocation de la fonction "time (NULL)" : elle renvoie l'heure actuelle connue de votre ordinateur (si votre ordinateur est faux sur ce point, le résultat sera également faux)
- type "tm" : une structure destinée à stocker des informations temporelles d'une manière plus pratique que celle utilisée par "time_t" ; elle contient des champs décrivant l'année, le mois, le jour, etc.
- Invocation de fonction "localtime ()" : elle convertit les données "time_t" en structure "tm" et renvoie un pointeur vers une structure interne de type "tm" ; la structure est remplie avec l'heure actuelle chaque fois que

la fonction est invoquée ; comme la fonction renvoie un pointeur, nous devons le déréférencer (*) et le copier dans notre variable privée

• trois champs de structure "tm" sont intéressants pour nous:

  o tm_year : stocke le nombre d'années depuis 1900, donc vous devez ajouter 1900 pour obtenir l'année en cours
  o tm_mon : stocke le numéro du mois en cours, mais nous devons vous avertir: "0" signifie janvier (ajouter 1 pour obtenir la valeur utilisable)
  o tm_mday : stocke le numéro du jour actuel dans le mois - heureusement, il n'y a pas d'autres pièges, vous pouvez l'utiliser "tel quel"

Maintenant vous pouvez compléter l'extrait présenté ci-dessous - comme il devrait montrer la date actuelle, donc nous ne sommes pas en mesure de fournir une sortie de test.
Vérifiez-le vous-même.

```
#include <iostream>
#include <ctime>

struct Date {
 int year;
 int month;
 int day;
};

Date today(void) {

 // Insert your code here

}

int main(void) {
 Date t = today();
 cout << t.year << "-" << t.month << "-" << t.day << endl;
 return 0;
}
```

3.4.7(6) Les nombres premiers - comment les trouver ?

Objectifs

- familiariser l'élève avec des notions et des algorithmes classiques,
- améliorer les compétences de l'étudiant dans la définition et l'utilisation des fonctions.

Scenario

Un nombre naturel est premier s'il est supérieur à 1 et n'a pas de diviseur autre que 1 et lui-même
Compliqué ? Pas du tout. Regardez : 8 n'est pas un nombre premier car vous pouvez le diviser par 2 et 4 (nous ne pouvons pas utiliser de diviseurs égaux à 1 et 8 car la définition l'interdit). D'autre part, 7 est un nombre premier car nous ne pouvons pas trouver de diviseurs légaux pour cela.
Votre tâche consiste à écrire une fonction en vérifiant si un nombre est premier ou non
La fonction:

- s'appelle "isPrime"
- prend un argument int (la valeur à vérifier)
- retourne "vrai" si l'argument est un nombre premier ou "faux" sinon
- devrait être muette

Astuce : essayez de diviser l'argument par toutes les valeurs suivantes (à partir de 2) et vérifiez le reste - si c'est zéro, votre nombre ne peut pas être un nombre premier ; réfléchissez bien quand vous devriez arrêter le processus et vérifier si vous pouvez utiliser la fonction "sqrt ()" déjà connue à cette fin.
Complétez le code présenté ci-dessous
Exécutez votre code et vérifiez si votre sortie est la même que la nôtre.

```
#include <iostream>
#include <cmath>

using namespace std;

bool isPrime(int num) {

 // Insert your code here
}

int main(void) {
 for(int i = 0; i <= 21; i++)
  if(isPrime(i))
   cout << i << " ";
 cout << endl;
 return 0;
}
```

Example output

```
2 3 5 7 11 13 17 19
```

3.6.3(1) Modification de la valeur des arguments de fonction - comment le faire ?

Objectifs

- concept de passer des arguments par référence,
- utiliser la fonction pour modifier ses arguments.

Scenario

Nous avons besoin d'une fonction très spécifique et nous en avons besoin le plus tôt possible. C'est ainsi que nous l'imaginons :

• son nom est "increment"
• elle retourne void
• lorsqu'elle est invoquée avec un argument (une variable int), elle incrémente simplement la variable de 1
• lorsqu'elle est invoquée avec deux arguments (une variable int et une expression int), elle incrémente la variable par la valeur de l'expression

Désolé, nous ne pouvons pas vous montrer un en-tête de cette fonction - ce serait trop facile. Nous allons seulement vous montrer comment nous invoquons la fonction - le reste vous appartient.
Complétez le code présenté ci-dessous.
Exécutez votre code et vérifiez si votre sortie est la même que la nôtre.

```
#include <iostream>

using namespace std;

// Insert your function here

int main(void) {
 int var = 0;
 for(int i = 0; i < 10; i++)
  if(i % 2)
   increment(var);
  else
   increment(var,i);
 cout << var << endl;
 return 0;
}
```

Example output

```
25
```

3.8.7(1) Fonctions de surcharge

Objectifs

- améliorer les compétences de l'étudiant en utilisant des mécanismes de «passage par référence»,
- familiariser l'étudiant avec les concepts de la fonction de surcharge.

Scenario

Désolé de vous avoir dérangé, mais nous devons vous dire que la fonction que vous avez implémentée la dernière fois doit être complétée par une autre qui effectue (plus ou moins) les mêmes actions mais en utilisant des types de données différents. Nous ne pouvons rien dire de plus, mais nous sommes sûrs que vous pourrez déduire toutes nos intentions et besoins du code présenté ci-dessous.
Complétez le code et exécutez-le pour vérifier si votre sortie est la même que la nôtre.

```
#include <iostream>
#include <cmath>
using namespace std;

// Insert your functions here

int main(void) {

 int intvar = 0;
 float floatvar = 1.5;

 for(int i = 0; i < 10; i++)
  if(i % 2) {
   increment(intvar);
   increment(floatvar, sqrt(intvar));
  }
  else {
   increment(intvar,i);
   increment(floatvar);
  }
 cout << intvar * floatvar << endl;
 return 0;
}
```

Example output

```
537.5
```

3.10.8(1) En utilisant des valeurs pseudo-aléatoires - une petite loterie

Objectifs

- améliorer les compétences de l'élève en utilisant les fonctions standard de la bibliothèque,
- améliorer les compétences de l'élève en utilisant des vecteurs,
- familiariser l'étudiant avec le concept de génération et d'utilisation de nombres pseudo-aléatoires.

Scenario

Ne pensez pas que nous voulons vous encourager à jouer - rien ne pourrait être plus faux. Nous voulons seulement que vous écriviez un code qui essaie de "prédire" (notez les guillemets) les nombres pour un jeu de loterie.

Il y a beaucoup de loteries différentes donc votre programme devrait être flexible. Il faudra connaître deux paramètres de base : combien de balles sont mises dans la machine et combien d'entre elles sont tirées. Par conséquent, votre code doit entrer deux valeurs int reflétant ces restrictions.

Ensuite, le programme devrait "dessiner" (notez les citations à nouveau) le nombre requis de balles. Pour simuler le processus de dessin, nous utiliserons ce qu'on appelle le "générateur de nombres pseudo-aléatoires" - un algorithme produisant une série de nombres se comportant comme s'ils étaient "dessinés", bien que les valeurs soient strictement déterministes et (tristes mais vraies) complètement prévisible.

Voici un petit code montrant comment l'utiliser :

```
#include <iostream>
#include <cstdlib>
#include <ctime>

using namespace std;

int main(void) {
 int max = 10;

 srand(time(NULL));
 int rnd = rand() % max + 1;
 cout << rnd << endl;
 return 0;
}
```

- Le fichier d'en-tête "cstdlib" est nécessaire car il fournit deux fonctions: srand () et rand ()
- Le fichier d'en-tête "ctime" est nécessaire parce que nous allons utiliser la fonction time ()
- La fonction "srand ()" initialise le générateur - le passage de la valeur d'horloge actuelle à la fonction rend le comportement du générateur plus aléatoire car la valeur de l'horloge est plutôt imprévisible
- invoquer la fonction "rand ()" nous donne un nombre pseudo-aléatoire de la plage [0 - RAND_MAX] (notez les crochets : ils indiquent que 0 et RAND_MAX sont également inclus dans la plage) ; La valeur réelle du symbole RAND_MAX dépend de l'implémentation et nous n'avons pas besoin de le connaître car nous voulons "normaliser" la valeur retournée à notre propre plage: [1..max] ; analyser notre code avec soin et devinez comment nous l'avons fait
- compilez le code et exécutez-le plusieurs fois - vous verrez qu'il sort un nombre qui semble être "aléatoire" (en d'autres termes: vous ne serez pas en mesure de le prédire)

Et voici le code que vous devez remplir. Nous voulons qu'il se comporte de la manière suivante :

- il devrait entrer deux valeurs int (vous obtenez cette partie gratuitement)
- Ensuite, il devrait "dessiner" autant de balles que l'utilisateur le souhaite mais rappelez-vous - vous ne pouvez pas utiliser le même nombre plus d'une fois !
- cela signifie que vous devez créer un tableau pour tous les numéros déjà dessinés (utilisez l'opérateur "new" !) Et vérifiez si le nouveau numéro n'a pas déjà été utilisé
- Enfin, vous devez afficher vos numéros et libérer la mémoire précédemment allouée (en utilisant l'opérateur delete [])

Testez votre code avec soin ! Ce n'est pas un jeu !

```
#include <iostream>
#include <cstdlib>
#include <ctime>

using namespace std;

int main(void) {
 int maxball;
 int ballsno;

 cout << "Max ball number? ";
 cin >> maxball;
 cout << "How many balls? ";
 cin >> ballsno;
 srand(time(NULL));

 // Insert your code here

 return 0;
}
```

3.10.8(2) Données dynamiques - comment l'obtenir et comment s'en débarrasser

Objectifs

- assigner des portions de données à new et les supprimer avec les opérateurs delete,
- le concept de collection dynamique de données.

Scenario

Regardez le code présenté ci-dessous - c'est un squelette d'un programme opérant sur la collecte dynamique de données. Son idée est d'utiliser une structure contenant deux champs : le premier stocke le nombre d'éléments dans les collections, le dernier est la collection actuelle (un vecteur alloué dynamiquement de int). Comme vous le voyez, la collection est remplie avec la quantité requise de données pseudo-aléatoires. Malheureusement, le programme nécessite l'achèvement comme la fonction la plus importante, destinée à ajouter des éléments à la collection encore vide. Êtes-vous prêt à relever le défi ?

Voici ce que nous attendons de la fonction :

- si la collection est vide, elle doit allouer un vecteur à un élément et y stocker une nouvelle valeur
- si la collection n'est pas vide, il faut allouer un nouveau vecteur avec une longueur supérieure de 1 au vecteur courant, puis copier tous les éléments de l'ancien vecteur dans le nouveau, ajouter une nouvelle valeur au nouveau vecteur et enfin libérer le vieux vecteur

Exécutez le code plusieurs fois et assurez-vous que tout fonctionne comme il se doit.

```
#include <iostream>
#include <cstdlib>
#include <ctime>

using namespace std;

struct Collection {
 int  elno;
 int *elements;
};

void AddToCollection(Collection &col, int element) {

 // Insert your code here

}

void PrintCollection(Collection col) {
 cout << "[ ";
 for(int i = 0; i < col.elno; i++)
  cout << col.elements[i] << " ";
 cout << "]" << endl;
}

int main(void) {
 Collection collection = { 0, NULL };

 int elems;
 cout << "How many elements? ";
 cin >> elems;
 srand(time(NULL));
 for(int i = 0; i < elems; i++)
  AddToCollection(collection, rand() % 100 + 1);
 PrintCollection(collection);
 delete[] collection.elements;
 return 0;
}
```