Екзамінаційний білет № 1

1. Введення-виведення для стандартного Сі на Сі++. Використання опцій для різного типу виводу та точності вхідних і вихідних параметрів Стандартна бібліотека шаблонів. Пакет vector
2. Задача: Реалізувати клас для роботи з рядками на базі масиву символів (члени класу – масив символів та його довжина)
   1. Введення-виведення для стандартного Сі на Сі++.

Засоби введення виведення в Сі. Загальні положення.

Мова Сі є фундаментом С + +. При цьому С + + підтримує всю файлову систему Сі. Тому при використанні С-коду в С + + немає необхідності міняти процедури введення-виведення. Хоча під час написання програм на С + + зазвичай більш зручно використовувати саме систему С + +. Це стосується, зокрема, і використання "iostream.h" замість "stdio.h", які реалізують введення-виведення. Вивчимо файловий ввід-висновок у мові Сі. Тим більше, що це саме по собі дуже цікаво і дуже важливо для розуміння "потоків" і "файлів" як у Сі, так і в С + +.

* 1. Використання опцій для різного типу виводу та точності вхідних і вихідних параметрів Стандартна бібліотека шаблонів.

У мові програмування C поняття типів даних відносять до великої системи оголошення змінних різних типів. Сама мова надає базові арифметичні типи, а також синтаксис для створення масивів.

Дані різного типу зберігаються і обробляються по-різному. У будь-якій алгоритмічній мові кожна константа, змінна, результат обчислення виразу або функції повинні мати певний тип. Тип даних визначає:

внутрішнє представлення даних в пам'яті комп'ютера;

область значень, які можуть приймати величини цього типу;

операції і функції, які можна застосовувати до величин цього типу.

Виходячи з цих характеристик, програміст вибирає тип кожної величини, використовуваної в програмі для подання реальних об'єктів. Обов'язковий опис типу дозволяє компілятору проводити перевірку допустимості різних конструкцій програми. Від типу величини залежать машинні команди, які будуть використовуватися для обробки даних.

Всі типи мови C можна розділити на основні і складові. У мові C визначено шість основних типів даних для представлення цілих, дійсних, символьних і логічних величин. На основі цих типів програміст може вводити опис складових типів. До них належать масиви, перерахування, функції, структури, посилання, визначники, об'єднання і класи. Основні типи даних часто називають арифметичними, тому що їх можна використовувати в арифметичних операціях. Для опису основних типів мови C використовують такі службові слова:

int (цілий);

char (символьний);

bool (логічний);

float (дійсний);

double (дійсний з подвійною точністю);

void (порожній, не має значення).

Short

Кожна змінна має певний тип. І цей тип визначає, які значення може мати змінна, які операції з нею можна робити і скільки байт в пам'яті вона буде займати.

У мові C визначені такі базові типи даних:

**bool:** логічний тип. Може приймати одну з двох значень true (істина) і false (брехня). Кількість пам'яті для цього типу точно не визначена.

**char:** представляє один символ в кодуванні ASCII.

**signed char і unsigned char**: представляє один символ.

**wchar\_t**: представляє розширений символ.

**char16\_t і char32\_t:** представляє один символ в кодуванні Unicode.

**int:** представляє ціле число.

**short і unsigned short**: також представляють ціле число.

**unsigned int:** представляє позитивне ціле число.

**long, long long і unsigned long long:** також представляють ціле число.

**float, double і long double**: представляють дійсне число ординарної точності з рухомою комою.

**void:** тип без значення.

* 1. Стандартна бібліотека шаблонів.

Стандартна бібліотека шаблонів (Standard Template Library; STL) — бібліотека для C++, що містить набір узгоджених узагальнених алгоритмів, контейнерів, засобів доступу до їхнього вмісту і різних допоміжних функцій.

Стандартна бібліотека шаблонів до включення в стандарт C++ була сторонньою розробкою, на початку — фірми HP, а потім SGI. Стандарт мови не називає її «STL», оскільки ця бібліотека стала невід'ємною частиною мови, проте багато людей досі використовують цю назву, щоб відрізняти її від решти частини стандартної бібліотеки (потоки вводу/виводу (iostream), підрозділ Сі тощо). У бібліотеці виділяють чотири основнi компоненти:

Контейнер (container) — зберігання набору об'єктів в пам'яті.

Ітератор (iterator) — забезпечення засобів послідовного доступу до вмісту контейнера.

Алгоритм (algorithm) — визначення обчислювальної процедури.

Функціональний об'єкт (functor) — заховання функції в об'єкті для використання іншими компонентами.

Розділення дозволяє зменшити кількість компонентів. Наприклад, замість написання окремої функції пошуку елементу для кожного типу контейнера забезпечується єдина версія, яка працює з кожним з них, поки дотримуються основні вимоги.

Контейнери бібліотеки STL можна розділити на чотири категорії: послідовні, асоціативні, контейнери-адаптери і псевдоконтейнери.

У бібліотеці STL для доступу до елементів як посередник використовується узагальнена абстракція, що іменується ітератором. Кожен контейнер підтримує «свій» вид ітератора, який є «модернізованим» інтелектуальним вказівником, що «знає» як отримати доступ до елементів конкретного контейнера. Стандарт C визначає п'ять категорій ітераторів, описаних в наступній таблиці: Вхідні, Вихідні, Однонаправлені, Двонаправлені й Довільного доступу.

* 1. Пакет vector

Класс вектора стандартной библиотеки C++ — это шаблон класса для контейнеров последовательностей. Вектор хранит элементы заданного типа в линейном расположении и обеспечивает быстрый случайный доступ к любому элементу. Вектор является предпочтительным контейнером для последовательности, если производительность случайного доступа находится на уровне "Премиум".

template <class Type, class Allocator = allocator<Type>>

class vector

Параметры: Type

Тип данных элементов, сохраняемых в векторе: Allocator

Тип, представляющий сохраненный объект распределителя, содержащий сведения о распределении и отмене распределения памяти для вектора. Этот аргумент является необязательным, и значением по умолчанию является allocator<Type>.

| **Имя** | **Описание** |
| --- | --- |
| [assign](https://docs.microsoft.com/ru-ru/cpp/standard-library/vector-class?view=msvc-170#assign) | Удаляет вектор и копирует указанные элементы в пустой вектор. |
| [at](https://docs.microsoft.com/ru-ru/cpp/standard-library/vector-class?view=msvc-170#at) | Возвращает ссылку на элемент в заданном положении в векторе. |
| [back](https://docs.microsoft.com/ru-ru/cpp/standard-library/vector-class?view=msvc-170#back) | Возвращает ссылку на последний элемент вектора. |
| [begin](https://docs.microsoft.com/ru-ru/cpp/standard-library/vector-class?view=msvc-170#begin) | Возвращает итератор произвольного доступа, указывающий на первый элемент в векторе. |
| [capacity](https://docs.microsoft.com/ru-ru/cpp/standard-library/vector-class?view=msvc-170#capacity) | Возвращает число элементов, которое вектор может содержать без выделения дополнительного пространства. |
| [cbegin](https://docs.microsoft.com/ru-ru/cpp/standard-library/vector-class?view=msvc-170#cbegin) | Возвращает постоянный итератор произвольного доступа, указывающий на первый элемент в векторе. |
| [cend](https://docs.microsoft.com/ru-ru/cpp/standard-library/vector-class?view=msvc-170#cend) | Возвращает константный итератор произвольного доступа, указывающий на позицию, следующую за концом вектора. |
| [crbegin](https://docs.microsoft.com/ru-ru/cpp/standard-library/vector-class?view=msvc-170#crbegin) | Возвращает константный итератор, который указывает на первый элемент в обратном векторе. |
| [crend](https://docs.microsoft.com/ru-ru/cpp/standard-library/vector-class?view=msvc-170#crend) | Возвращает константный итератор, который указывает на последний элемент в обратном векторе. |
| [clear](https://docs.microsoft.com/ru-ru/cpp/standard-library/vector-class?view=msvc-170#clear) | Очищает элементы вектора. |
| [data](https://docs.microsoft.com/ru-ru/cpp/standard-library/vector-class?view=msvc-170#data) | Возвращает указатель на первый элемент в векторе. |
| [emplace](https://docs.microsoft.com/ru-ru/cpp/standard-library/vector-class?view=msvc-170#emplace) | Вставляет элемент, созданный на месте, в указанное положение в векторе. |
| [emplace\_back](https://docs.microsoft.com/ru-ru/cpp/standard-library/vector-class?view=msvc-170#emplace_back) | Добавляет элемент, созданный на месте, в конец вектора. |
| [empty](https://docs.microsoft.com/ru-ru/cpp/standard-library/vector-class?view=msvc-170#empty) | Проверяет, пуст ли контейнер вектора. |
| [end](https://docs.microsoft.com/ru-ru/cpp/standard-library/vector-class?view=msvc-170#end) | Возвращает итератор произвольного доступа, который указывает на конец вектора. |
| [erase](https://docs.microsoft.com/ru-ru/cpp/standard-library/vector-class?view=msvc-170#erase) | Удаляет элемент или диапазон элементов в векторе из заданных позиций. |
| [front](https://docs.microsoft.com/ru-ru/cpp/standard-library/vector-class?view=msvc-170#front) | Возвращает ссылку на первый элемент в векторе. |
| [get\_allocator](https://docs.microsoft.com/ru-ru/cpp/standard-library/vector-class?view=msvc-170#get_allocator) | Возвращает объект классу allocator, используемому вектором. |
| [insert](https://docs.microsoft.com/ru-ru/cpp/standard-library/vector-class?view=msvc-170#insert) | Вставляет элемент или несколько элементов в вектор по заданной позиции. |
| [max\_size](https://docs.microsoft.com/ru-ru/cpp/standard-library/vector-class?view=msvc-170#max_size) | Возвращает максимальную длину вектора. |
| [pop\_back](https://docs.microsoft.com/ru-ru/cpp/standard-library/vector-class?view=msvc-170#pop_back) | Удаляет элемент в конце вектора. |
| [push\_back](https://docs.microsoft.com/ru-ru/cpp/standard-library/vector-class?view=msvc-170#push_back) | Добавляет элемент в конец вектора. |
| [rbegin](https://docs.microsoft.com/ru-ru/cpp/standard-library/vector-class?view=msvc-170#rbegin) | Возвращает итератор, указывающий на первый элемент в обратном векторе. |
| [rend](https://docs.microsoft.com/ru-ru/cpp/standard-library/vector-class?view=msvc-170#rend) | Возвращает итератор, который указывает на последний элемент в обратном векторе. |
| [reserve](https://docs.microsoft.com/ru-ru/cpp/standard-library/vector-class?view=msvc-170#reserve) | Резервирует минимальную длину хранилища для объекта вектора. |
| [resize](https://docs.microsoft.com/ru-ru/cpp/standard-library/vector-class?view=msvc-170#resize) | Определяет новый размер вектора. |
| [shrink\_to\_fit](https://docs.microsoft.com/ru-ru/cpp/standard-library/vector-class?view=msvc-170#shrink_to_fit) | Удаляет лишнюю емкость. |
| [size](https://docs.microsoft.com/ru-ru/cpp/standard-library/vector-class?view=msvc-170#size) | Возвращает количество элементов в векторе. |
| [swap](https://docs.microsoft.com/ru-ru/cpp/standard-library/vector-class?view=msvc-170#swap) | Меняет местами элементы двух векторов. |

**Функции**

**Операторы**

| **Имя** | **Описание** |
| --- | --- |
| [operator[]](https://docs.microsoft.com/ru-ru/cpp/standard-library/vector-class?view=msvc-170#op_at) | Возвращает ссылку на элемент вектора в указанной позиции. |
| [operator=](https://docs.microsoft.com/ru-ru/cpp/standard-library/vector-class?view=msvc-170#op_eq) | Заменяет элементы вектора копией другого вектора. |