3.5 Синтаксис использования конструкций if. . . else, switch, for, while, do... while. Использование команд break, continue, return

Использование конструкции if... else:

```
if (bool condition) {
    statement1;
}
else {
    statement2;
}
```

Использование конструкции switch:

```
switch(expr) {
    case 1:
        statement1;
        break;
    case 2:
        statement2;
        break;
    ...
    default:
        defstat;
        break;
}
```

Использование конструкции for:

```
for (decl or expr; bool-expr; expr) {
        statements;
        continue;
    }
```

Использование конструкции while:

```
    while (condition) {
        expr;
    }
```

Использование конструкции do... while:

```
do {
    expr;
} while (condition);
```

Команда break используется для выхода из цикла, continue — для перехода на следующую итерацию цикла, return — для выхода из функции.

3.6 Разница между объявлением и определением. Синтаксис объявления и определения переменных, функций, классов и структур, алиасов для типов (с помощью слова using). Правило одного определения (ODR) для функций и для переменных, объяснение его смысла.

Declaration (Объявление). При объявлении чего-то всё, что вы делаете, это говорите компилятору, что есть что-то с определенным именем и определенного типа.)

Definition (Определение). Определение чего-то означает предоставление всей необходимой информации для создания этого целиком.)

Таким образом, разница в том, что после определения чего-то объект обязательно жизнеспособен. Можно объявлять, но не определять - например функции:

```
int f(); //declaration but not definition
int g(){ //definition
    f();
}
int f(){ //definition
    g();
}
```

Что можно объявлять?

1. Переменные
2. Функции
3. Классы
4. Структуры
5. Union
6. Namespace
7. Alias (Другое название для чего-нибудь, т.е. псевдоним (см. пример с using)

Синтаксис объявления и определения:
type id [=init]; - объявление переменной [определение].
type func_id($type_1 id_1, ..., type_n id_n$); - объявление функции (добавляем фигурные скобки для определения). Пример объявления:

```
namespace N {}
typedef vvi vector<vector<int>>;
using vvi =  vector<vector<int>>;
```

One Definition Rule. Оно гласит много чего, в частности функции и переменные должны быть определены ровно один раз. Формально - Only one definition of any variable, function, class type, enumeration type, concept (since C++20) or template is allowed in any one translation unit (some of these may have multiple declarations, but only one definition is allowed).)

3.7 Разница между стековой (автоматической) и динамической памятью. Динамическое выделение памяти с помощью оператора new (в стандартной форме). Освобождение динамической памяти. Проблема утечек памяти. Проблема двойного удаления.

Автоматическая память (стек)

Непосредственное управление автоматической памятью — выделение памяти под локальные объекты при их создании и освобождение занимаемой объектами памяти при их разрушении — осуществляется компилятором. Собственно, по этой причине память и называют автоматической.

Период времени от момента создания объекта до момента его разрушения, т. е. период времени, в течение которого под объект выделена память, называют временем жизни объекта.

Время жизни локальных объектов определяется областью видимости их имён. Область видимости локального имени начинается с места его объявления и заканчивается в конце блока, в котором это имя объявлено. Область видимости аргументов функции ограничивается телом функции, т. е. считается, что имена аргументов объявлены в самом внешнем блоке функции.

Статическая память

Когда программа запускается, ОС выделяет под неё память. В этой памяти есть три раздела (и не только): data, text (секция кода), stack (4 мБ). В data хранятся статические переменные - их время жизни определено сроком жизни программы.

Динамическая память

Допустим, мы хотим выделить в runtime дополнительную память. Для этой цели подходит оператор new: type* ptr = new type(size). Такое выделение памяти будет существовать до тех пор, пока мы явно не укажем, что хотим его очистить, для этого существует оператор delete. Эти операторы – достаточно низкоуровневый инструмент.

Связанные проблемы: утечки памяти (утерян указатель на выделенную память или память не была очищена), очищение памяти, которая не была выделена (когда по ошибке очистка памяти делается по иной области памяти), проблема двойного удаления (в программе по одному и тому же указателю дважды очищается память)