Язык Си

Типы данных

Язык Си является языком со статической типизацией, т.е. тип переменной известен на этапе компиляции и не меняется в процессе работы. Компилятор берет на себя проверку типов, снимая с программиста заботу о проверке типов(в отличии от динамической типизации где на этапе исполнения типы переменных могут не совпадать и произойдет ошибка процесса исполнения). В современных языках используется прием вычисление типа на этапе компиляции. Если переменная обозначена специальным типом, то в процесс екомпиляции будет расичтан тип переменной и назначен ей. Такой прием применяется в новых стандартах языка `C++`, вычисляемым типом является `auto`.

Также, в языка Си применется слабая типизация, т.е. есть возможность привести один тип к другому, без проверки. Данный прием называется приведением типов. Эта операция считается опасной, и может привести к ошибкам на этапе исполнения программы, если типы не совпадают. Например при не совпадении типов указателей, при итерировании, можно выйти за границу массива и повредить память или при приведении типа `long long` к типу `int` можно потерять часть данных которые выходят за размер переменной. В современных компиляторах, преобразование типов, на стадии семантического анализа, производится попытка предположить возможность появления ошибки. Приведение типов записывается как

`(int *)v` - что означает приведение переменной `v` к типу `int *` и назвается `C-style` преобразование типов. Данный вид записи встречается во многих языках, нгапример в `Java`, `C#` - хотя в нем есть более удачные конструкции преобразования. Из-за приведения типов, язык не может считаться типобезопасным. В `C++`, такой стиль считается устаревшим и выдается предупреждение на этапе компиляции и предлагается использовать `static_cast`, `const_cast`, `reinterpret_cast`, `dynamic_cast` - что является более безопасным и контролируемым компилятором.

Немного отступая, можно рассказать еще о так называемой “утиной типизации” - это такой вид типизации когда интерфейс явно не реализуется, а реализуются его методы. Данные подход применяется в языке `GoLang`.

Си также не грешит “каламбуром” типизации. Это когда обходится проверка системы типов компилятора, для выполнения определенных задач. Одним из примеров может быть интерфейс сокетов. Функция `bind` имеет следующее описание:

```
int bind(int sockfd, struct sockaddr *my_addr, socklen_t addr
len);
```

Вызов функции происходит примерно следующим образом

```
struct sockaddr_in sa = {0};
int sockfd = ...;
```

```
sa.sin_family = AF_INET;
sa.sin_port = htons(port);
bind(sockfd, (struct sockaddr *)&sa, sizeof sa);
```

Применяется за основу тот факт, что в языке указатель на `struct sockaddr_in` может беспрепятственно преобразовываться в указатель на `struct sockaddr`, а также что оба структурных типа частично совпадают по организации представления в памяти. Следовательно, указатель на поле `v->sin_family` (где `v` имеет тип `struct sockaddr*`) на самом деле будет указывать на поле `sa.sin_family` (где `sa` имеет тип `struct sockaddr_in`). Другими словами, библиотека использует каламбур типизации для реализации примитивной формы наследования.

Такой же подъод можно встретить в `Windows API`.