Отладка программ. Использование gdb. Классификация допускаемых при написании кода ошибок.

Ошибки в коде – это непредвиденная ситуация, которая может возникнуть во время выполнения программы и требует специальной обработки для корректной работы программы. Не обработанные ошибки могут приводить к **непредсказуемому поведению программы**, включая **сбои**, **повреждение данных** и **скрытые уязвимости** в безопасности.
Типы ошибок:

- Синтаксические ошибки
- Семантические ошибки
- Логические ошибки
- Ошибки времени выполнения
- Ошибки работы с памятью
  Синтаксическая ошибка – опечатка, не знание синтаксиса, которая тормозит компиляцию. Это наиболее простые ошибки для обнаружения. Они возникают из-за **нарушения правил языка программирования**.

```
printf("Hello, World!\n")   // <- нет ;
return 0;
```

Семантическая ошибка – программа работает, но не так, как задумано.

```
if (num_purchases >= 10)
    printf("Вы получили скидку!"); // если условие верно, выполняется только
эта строка, так как нет фигурных скобок
    subtotal = subtotal * 0.9; // эта строка будет выполятся всегда
```

Логическая ошибка – всё «работает», но результат неверный. Ошибка в алгоритме, которая приводит к некорректному выводу.

```
int i, j;
j=(++i) * 2 + --i;
```

Ошибки времени выполнения – программа падает во время работы. Частые причины: деление на ноль, выход за границы массива, доступ к освобожденной памяти, бесконечные циклы.

Ошибки работы с памятью:

- Утечки памяти (memory leaks)
- Использование освобождённой памяти (use-after-free)
- Чтение неинициализированной памяти (uninit read)
- Переполнение буфера (buffer overflow)
- Двойное освобождение (double free)
- Доступ за границами массива
- Разыменование NULL-указателя

Способы решения проблем
Отладка (debugging) — это процесс выявления, анализа и устранения дефектов программы, которые приводят к её некорректному поведению или аварийному завершению.
Основные инструменты:

- **printf debugging** — вы вручную добавляете вызовы **printf()** в ключевые точки кода, чтобы **наблюдать** за состоянием переменных, порядком выполнения, а также **для поиска источника ошибок**.
- **GNU Debugger (GDB)** — это мощный инструмент отладки с открытым исходным кодом , разработанный проектом GNU. Он позволяет анализировать поведение программы на низком уровне , что особенно важно при системном программировании.
- **Valgrind (Memcheck)** — помогает находить ошибки, связанные с работой с памятью.
- **strace/ltrace** — это утилита командной строки в Unix-подобных системах, которая перехватывает и записывает все системные вызовы , сделанные процессом, а также все сигналы , полученные этим процессом.

GNU Debugger (GDB) предназначен для:

- Пошагового выполнения программы
- Установки точек останова (breakpoints)
- Просмотра состояния памяти и регистров
- Анализа падений программы (например, segfault)
- Работы с многопоточными и low-level программами
  Для компиляции нужен флаг **-g**
  Этот флаг добавляет в исполняемый файл информацию о: именах переменных,

номерах строк, именах функций, типах данных. Без этого **GDB будет работать**, но без возможности просматривать исходный код и переменные.

```
gcc -g -o my_program my_program.c
```

Запуск отладчика:

```
gdb ./my_program
```

Основные команды:

run(**r**)	Запуск программы
break(**b**)	Установка точки останова (по имени функции или номеру строки)
step(**s**)	Выполнить следующую строку (входить внутрь функций)
next(**n**)	Выполнить следующую строку (не входить в функции)
continue(**c**)	Продолжить выполнение до следующей точки останова

display	Добавить выражение из списка автоматического отображения
undisplay	Удалить выражение из списка автоматического отображения
print(**p**)	Вывести значение переменной или выражения
info registers	Показать содержимое регистров процессора
backtrace(**bt**)	Показать трассировку стека вызовов
Disassemble	Показать машинный код функции
quit(**q**)	Выйти из GDB