ЎЗБЕКИСТОН РЕСПУБЛИКАСИ
ОЛИЙ ВА ЎРТА МАХСУС ТАЪЛИМ ВАЗИРЛИГИ
БУХОРО ДАВЛАТ УНИВЕРСИТЕТИ
АХБОРОТ ТЕХНОЛОГИЯЛАРИ ФАКУЛЬТЕТИ

АМАЛИЙ МАТЕМАТИКА ВА АХБОРОТ ТЕХНОЛОГИЯЛАРИНИНГ ЗАМОНАВИЙ МУАММОЛАРИ

ХАЛҚАРО МИҚЁСИДАГИ ИЛМИЙ-АМАЛИЙ АНЖУМАН

МАТЕРИАЛЛАРИ

2021 йил, 15-апрель

Бухоро – 2021

ТАШКИЛИЙ ҚЎМИТА

Раис: Хамидов О.Х., БухДУ ректори, профессор
Раис ўринбосари: Қаххоров О.С., БухДУ проректори, доцент
Ташкилий қўмита аъзолари:

Жўраев А.Т.	БухДУ, проректори, доцент
Рашидов Ў.У.	БухДУ, проректори
Зарипов Г.Т.	БухДУ, доцент
Эшанкулов Х.И.	БухДУ, декан, т.ф.ф.д., (PhD)
Жалолов О.И.	БухДУ, кафедра мудири, доцент
Сайидова Н.С.	БухДУ, кафедра мудири, доцент
Жумаев Ж.	БухДУ, доцент
Болтаев Т.Б.	БухДУ, доцент
Зарипова Г.К.	БухДУ, доцент
Рустамов Ҳ.Ш.	БухДУ, доцент
Хаятов Х.У.	БухДУ, катта ўқитувчи
Жўраев З.Ш.	БухДУ, катта ўқитувчи
Атаева Г.И.	БухДУ, катта ўқитувчи
Турдиева Г.С.	БухДУ, катта ўқитувчи

ДАСТУРИЙ ҚЎМИТА

Арипов М.М.	ЎзМУ, профессор
Алоев Р.Ж.	ЎзМУ, профессор
Шадиметов Х.М	Тошкент давлат транспорт университети, профессор
Расулов А.С.	Жаҳон иқтисодиёти ва дипломатия университети, профессор
Равшанов Н.	ТАТУ ҳузуридаги АКТ илмий-инновацион марказ, лаборатория мудири, профессор
Солеев А.С.	СамДУ, профессор
Дурдиев Д.Қ.	БухДУ, профессор
Ҳаётов А.Р.	В.И.Романовский номидаги Математика институти, профессор
Мўминов Б.Б.	ТАТУ, профессор
Худойберганов М.У.	ЎзМУ, доцент
Жумаев Ж.	БухДУ, доцент
Болтаев Т.Б.	БухДУ, доцент
Эшанкулов Х.И.	БухДУ, т.ф.ф.д., (PhD)
Жалолов О.И.	БухДУ, доцент
Сайидова Н.С.	БухДУ, доцент
Расулов Т.Ҳ	БухДУ, доцент

КОНФЕРЕНЦИЯ КОТИБЛАРИ

Атамурадов Ж.Ж., Эргашев А.А. Қосимов Ф.Ф., Ҳазратов Ф.Ҳ., Зарипов Н.Н., Ибрагимов С.И., Назаров Ш.Э.

Тўплам Ўзбекистон Республикаси Вазирлар Маҳкамасининг 2021 йил 2 мартдаги 78-ф-сонли фармоиши билан тасдиқланган Ўзбекистон Республикасида 2021 йилда халқаро ва республика миқёсидаги ўтказиладиган илмий ва илмий-техник тадбирлар режасида белгиланган тадбирларнинг бажарилиши мақсадида 2021 йил 15 апрель куни Бухоро давлат университети Ахборот технологиялари факультетида “Амалий математика ва ахборот технологияларининг замонавий муаммолари” мавзусидаги халқаро илмий-амали анжуман материаллари асосида тузилди.

Масъул муҳаррир:
О.И.Жалолов, доцент
Тақризчилар:
Ж.Жумаев, доцент

меньше всего подвержено изменениям, поэтому оно является самым простым из всех. Например, программа-калькулятор для математических вычислений.

- **P-type (практический тип)** - это программное обеспечение с набором процедур. Это определяется тем, что именно могут делать процедуры. В этом программном обеспечении можно описать спецификации, но решение не сразу становится очевидным. Например, игровой софт.
- **E-type (встроенный)** - это программное обеспечение работает в соответствии с требованиями реальной среды. Это программное обеспечение имеет высокую степень развития, поскольку в реальных ситуациях происходят различные изменения в законах, налогах и т. д. Например, программное обеспечение для онлайн-торговли.

Литература

1. *Guckenheimer S.*, Perez J.I. Software Engineering with Microsoft Studio Team System.- Crawfordsville, USA: Adison-Wesley, 2006.-304 P.

2. 50. *Xteglio A., Begin M.E., Couvares P.. Ronchieri E., Takacs E.* ETICS: the International Software Engineering Service for the Grid - Jornal of Physics Conference Series 119. - 2008. - C.58-67.

СИСТЕМНОЕ ПРОГРАММИРОВАНИЕ НА C

[1]Атаева Гульсина Исроиловна, [2]Шокиров Обиджон Шухрат угли

[1]старший преподаватель кафедры информационных технологий БухГУ,

[2]студент 2-курса направления Прикладная математика и информатика.

Хотя определение системного программирования нечеткое, его можно описать как необходимость мыслить на уровне бита, байта, инструкции или цикла ЦП. Системное программирование также подразумевает высокие требования к производительности и надежности. Джо Даффи, технический директор Microsoft, представил на QCon New York стратегии системного программирования на C #.

Несколько уроков из выступления Джо были взяты из исследовательского проекта под названием Мидори. Проект был посвящен созданию операционной системы с нуля с использованием C #, что привело к новому пониманию конструкции компилятора, а также новым стратегиям для высокопроизводительного кода.

Использование управляемых языков для создания операционной системы дает возможность использовать функции безопасности C # на уровне памяти. Он предлагает решение против эксплойтов, основанных на памяти, таких как внедрение кода, из-за переполнения буфера или уязвимостей строки формата, поскольку среда выполнения заботится о проверке привязки и безопасности типов.

Генерация кода. Код может быть составлен либо загодя (AOT), или как раз вовремя (JIT). Преимущество JIT - быстрое время компиляции. С другой стороны, AOT дает лучший машинный код, поскольку компилятор может выполнять больше оптимизаций.

Некоторые оптимизации, сделанные компиляторами родных языков, традиционно были недоступны для управляемых языков. Общие причины обычно заключались в том, что оптимизация была либо слишком затратной с точки зрения вычислений, либо слишком сложной для выполнения в Jit-компиляторе. Это привело к тому, что C # имел плохую репутацию в сфере создания компактного, эффективного низкоуровневого кода. С помощью RyuJit недавно были реализованы следующие оптимизации:

- Встраивание (замена сайта вызова функции телом вызываемой функции)
- Анализ потокового графа и цикла
- Статическое одиночное присвоение (SSA) и нумерация глобальных значений
- Устранение общих подвыражений
- Копирование / распространение констант
- Устранение мертвого кода

- Анализ диапазона
- Девиртуализация
- Подъем инвариантного кода цикла
- SIMD и векторизация
- Общий доступ
- Распределение стека (работа в процессе)

Повышение производительности. Сборщик мусора в .NET является поколенческим, состоит из трех поколений. Некоторые программы анализа данных тратят более половины своего времени на сборку мусора, а это время, когда они не выполняют фактическую работу.

Один из способов повысить производительность - использовать структуры. Структуры улучшают производительность на следующих уровнях:
- Снижение нагрузки на сборщик мусора , поскольку структуры распределяются в стеке.
- Лучшая локализация памяти, повышение частоты попаданий в кеш.
- Меньшее общее использование памяти, избегая 8–16-байтовых накладных расходов объектов в 32-64-разрядных приложениях.

Одно предостережение относительно структур заключается в том, что их копирование может привести к memcpy при превышении определенного размера. Для оптимальной производительности структуры должны быть небольшими, менее 32/64 байта.

Некоторые функции C # 7 направлены на упрощение низкоуровневой оптимизации с использованием структур. Кортежи C # 7 являются структурами, а не предыдущей версией System.Tuple <>, которая является объектом. Ref return - еще одна функция для структур, где структура может быть возвращена из функции, избегая ее копирования.

Исключения **обработки ошибок** предназначены для исправимых ошибок. Однако многие ошибки не подлежат исправлению. Такие ошибки, как недопустимое приведение, переполнение стека и пустые ссылки, на самом деле являются ошибками. С другой стороны, сбои ввода-вывода и ошибки проверки следует ожидать и устранять их.

Восстановление после ошибок приводит к стратегии быстрого отказа. Fail fast - это механизм, включенный в .NET, где некоторые исключения, такие как StackOverflow, обходят обработчики исключений и приводят к сбою процесса. Это упрощает поиск ошибок, поскольку исключение не может быть обработано слишком универсальным обработчиком исключений. Команда Midori обнаружила, что у них соотношение исправимых ошибок (исключений) к числу ошибок (быстрое сбой) 1:10.

Коды ошибок. Коды ошибок, возможно, являются самой простой возможной моделью ошибки. Идея очень проста и даже не требует поддержки языка или среды выполнения. Функция просто возвращает значение, обычно целое число, чтобы указать успех или неудачу:

```
int foo() {
   // <try something here>
   if (failed) {
      return 1;
   }
   return 0;
}
```

Это типичный паттерн, где возврат `0` означает успех, а ненулевое значение означает неудачу. Вызывающий должен это проверить:

```
int err = foo();
if (err) {
   // Error!  Deal with it.
}
```

Большинство систем предлагают константы, представляющие набор кодов ошибок, а не магические числа. Могут быть, а могут и не быть функции, которые вы можете использовать для получения дополнительной информации о самой последней ошибке (например, `errno` в стандартном C и `GetLastError` в Win32). Код возврата на самом деле не является чем-то особенным в языке - это просто возвращаемое значение.

Литература

1. Макаров, А.В. Common Intermediate Language и системное программирование Microsoft . NET: Учебное пособие / А.В. Макаров. - М.: Бином, 2011. - 328 с.
2. Хаятов Х.У., Атаева Г.И., Хайдаров О.Р. Функции и элементы OPENGL, используемые для построения основных форм в C# // Universum: технические науки. №11(80), Часть 1, 2020. С. 43-46.
3. Ataeva Gulsina Israilovna, Shokirov Obidzhon. THE USE OF INTEGRATED TECHNOLOGIES IN THE EDUCATIONAL PROCESS// International Conference. BRIDGE TO SCIENCE: RESEARCH WORKS. December, 15, 2020, San Francisco, California, USA. Pp.97-99.
4. Атаева Г.И., Минич Л.С. Создание вывода скрипта Python// Вестник науки и образования. 2021. №1(104). Часть 2. С.12-15.
5. http://joeduffyblog.com/2016/02/07/the-error-model/
6. https://qconnewyork.com/

MS PROJECTDA YANGI LOYIHA YARATISH

Sayidova Nazokat Sayfullayevna, Jo`rayev Ilhom Isoqovich, Turayeva Mayram Hayit *qizi*

BuxDU Axborot texnologiyalari kafedrasi dotsenti, f.-m.f.n.

BuxDU Axborot texnologiyalari o`qituvchisi

BuxDU Kompyuter ilmlari va dasturlash yo`nalishi magistranti

MS Project dasturi tuzilishi jihatidan juda sodda bo`lib, u uchta subyekt bilan ishlaydi. Bular quyidagilar:vazifalar, manbalar, taqvim va ular orasidagi aloqalar. Aslida, bu ma'lumotlar bazasi, obyektlarni yaratish va tahrirlash uchun foydalanuvchi interfeysi va minimal, juda sodda avtomatlashtirish (loyiha kirish ma'lumotlariga javoban o'zini-o'zi bajaradi). Vazifa muddati, hajmi, tayinlangan resursi va juda ko'p turli xil xususiyatlarga ega. Agar o'rnatilgan xususiyatlar yetarli bo'lmasa, o'zingiznikini qo'shishingiz mumkin. Vazifalar bir-biri bilan turli xil munosabatlar bilan bog'liq bo'lishi mumkin. Dastur ko'plab tavsiflovchi xususiyatlarga ega, ammo eng muhimi qilish mumkin vazifalarni o'z vaqti belgilab qo`yiladi, buning uchun taqvim ishlatiladi.

Loyiha yordamida turli xil qarashlarni amalga oshirishga qodir filtrlar, guruhlar mavjud. Bundan tashqari, u ba'zi algoritmlardan qanday foydalanishni biladi tayinlangan resurslar mavjudligiga qarab vazifalar uchun boshlanish va tugash sanalarini hisoblash va vazifalar orasidagi bog'lanishni, reja tayyorlash uchun bizning oldimizda quyidagi texnik vazifalarni belgilab beradi:

1. Ushbu loyiha qancha vaqtni oladi?
2. Buning uchun qancha (va qanday) mutaxassislar kerak bo'ladi?
3. Ushbu loyiha uchun taxminiy mehnat xarajatlari qancha?

Buning uchun biz MS Project dasturida taxminiy loyiha rejasini tayyorlash kerak va bajarilishi kerak bo'lgan vazifalarni ketma-ket yozib qo'yish kerak bo`ladi.

Rejani tayyorlash bir necha bosqichda amalga oshiriladi:

1. Vazifalar ro'yxatini tayyorlash;
2. Vazifalar o'rtasidagi bog'liqlikni aniqlaymiz (qaysi vazifaning natijasi keyingisiga o'tish uchun talab qilinadi?);
3. Vazifalarni bajaruvchilarni tayinlash;
4. Resurslarni yuklashni tenglashtirish;
5. Nima bo'lganini muvozanatlash.