МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ГРАЖДАНСКОЙ АВИАЦИИ

А.А. Антонов

МЕТОДЫ ПРОГРАММИРОВАНИЯ

Учебно-методическое пособие
по проведению практических занятий

для студентов II курса
специальности 10.05.02
очной формы обучения

Москва · 2022

ФЕДЕРАЛЬНОЕ АГЕНТСТВО ВОЗДУШНОГО ТРАНСПОРТА
(РОСАВИАЦИЯ)

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ГРАЖДАНСКОЙ АВИАЦИИ» (МГТУ ГА)

Кафедра основ радиотехники и защиты информации

А.А. Антонов

МЕТОДЫ ПРОГРАММИРОВАНИЯ

Учебно-методическое пособие
по проведению практических занятий

для студентов II курса
специальности 10.05.02
очной формы обучения

Москва
ИД Академии Жуковского
2022

УДК 004.424
ББК 6Ф7.3
А72

Рецензент:
Петров В.И. – канд. техн. наук, доцент

Антонов А.А.
А72 Методы программирования [Текст] : учебно-методическое пособие по проведению практических занятий / А.А. Антонов. – М.: ИД Академии Жуковского, 2022. – 24 с.

Учебно-методическое пособие соответствует рабочей программе учебной дисциплины «Методы программирования» по специальности 10.05.02 для студентов II курса очной формы обучения.

В учебно-методическом пособии рассматриваются задания для практических занятий, получение практических навыков программирования на языке высокого уровня, применение существующих структур данных и алгоритмов. Рассматриваются методы разработки эффективного программного обеспечения.

Рассмотрено и одобрено на заседаниях кафедры 19.04.2022 г. и методического совета 21.04.2022 г.

УДК 004.424
ББК 6Ф7.3

В авторской редакции

Подписано в печать 13.07.2022 г.
Формат 60х84/16 Печ. л. 1,5 Усл. печ. л. 1,395
Заказ № 903/0603-УМП14 Тираж 30 экз.

Московский государственный технический университет ГА
125993, Москва, Кронштадтский бульвар, д. 20

Издательский дом Академии имени Н. Е. Жуковского
125167, Москва, 8-го Марта 4-я ул., д. 6А
Тел.: (495) 973-45-68
E-mail: zakaz@itsbook.ru

ОБЩИЕ МЕТОДИЧЕСКИЕ УКАЗАНИЯ

При подготовке к практическому занятию студенты должны:

уяснить цель и порядок проведения занятия;

изучить материалы, изложенные на лекциях и в рекомендуемой литературе.

На занятии студент должен иметь конспект лекций и данное пособие.

Практическое занятие начинается с опроса студентов по знанию теоретических положений изучаемого практического занятия, проверяются знания по представленным контрольным вопросам.

Далее студенты решают приведенные в пособии задания с последующим обсуждением полученных результатов. В случае дистанционного обучения оформляется отчет.

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

1. Павловская Т.А. C/C++. Программирование на языке высокого уровня. Учебник. – СПб.: Питер, 2009.

2. Подбельский В.В. Стандартный Си++. Учебное пособие. - М.: Финансы и статистика, 2008.

3. Петров В.И., Антонов А.А. Методы и средства программирования. Пособие по выполнению лабораторных работ. – М.: МГТУ ГА, 2010.

4. Матыюк С.П. Безопасность информационно-вычислительных систем воздушного транспорта. Учебное пособие. МГТУГА. 2020.

Практическое занятие № 1

Системы счисления

1. Цель занятия - получение практических навыков перевода чисел из одной системы счисления в другую, а также дополнительных кодов чисел.

2. Контрольные вопросы

1. Перевод чисел из одной системы счисления в другую.
2. Получение прямого, обратного и дополнительного кода.

3. Задание на практическое занятие

При подготовке к занятию необходимо изучить материалы лекций, и ответить на контрольные вопросы, используя рекомендованную литературу.

Цель занятия достигается путем получения навыков при решении представленных задач:

Задача № 1
Переведите числа из десятичной системы счисления
в двоичную, восьмеричную и шестнадцатеричную системы счисления.
Запишите ход решения.
а) 737; б) 92; в) 934,25; г) 413,5625; д) 100,94.

Задача № 2
Переведите числа в десятичную систему счисления.
Запишите ход решения.
а) 1110000010_2; б) 1000100_2; в) $110000100,001_2$;
г) $1001011111,00011_2$; д) $665,42_8$; е) $246,18_{16}$.

Задача № 3
Выполните сложение чисел.
Запишите ход решения.
а) $11110100_2+110100001_2$; б) $1101110_2+101001000_2$;
в) $1100110011,1_2+111000011,101_2$; г) $1455,04_8+203,3_8$;
д) $14E,8_{16}+184,3_{16}$.

Задача № 4
Выполните вычитание чисел.
Запишите ход решения.
а) $1000010101_2-100101000_2$; б) $1001011011_2-101001110_2$;
в) $111111011,101_2-100000010,01_2$; г) $341,2_8-275,2_8$; д) $249,5_{16}-EE,A_{16}$.

Задача № 5
Получить $[X]_{пр}$, $[X]_{обр}$, $[X]_{доп}$ следующих чисел:
x = - 0,1011; x = 0,0011;
x = - 1,0000; x = - 01111.
Записать ход решения.

Задача № 6
Вычислить $[X+Y]_{доп}$ и $[X-Y]_{доп}$ для чисел:
а) x = 0,1000 y = 0,1100; б) x = - 0,1100 y = 0,1011;
в) x = 0,1101 y = - 0,0010; г) x = - 0,1001 y = - 0,0110.
Записать ход решения.

4. Содержание отчета о практическом занятии

1. Титульный лист
2. Ответы на контрольные вопросы
3. Задачи
4. Решение
5. Вывод

Практическое занятие № 2

Программирование линейных алгоритмов

1. Цель занятия - получение практических навыков в решении задач на ЭВМ общего назначения, с использованием программ линейных алгоритмов.

2. Контрольные вопросы

1. Математическая формулировка задач.
2. Разработка линейных алгоритмов решения задач
3. Программирование линейных алгоритмов

3. Задание на практическое занятие

При подготовке к занятию необходимо изучить материалы лекций, и ответить на контрольные вопросы, используя рекомендованную литературу.

Цель занятия достигается путем получения навыков при решении представленных задач:

Задача №1.

Составить программу для расчёта расстояния d между двумя точками и углов α, β, γ, наклона прямой к осям x,y,z .

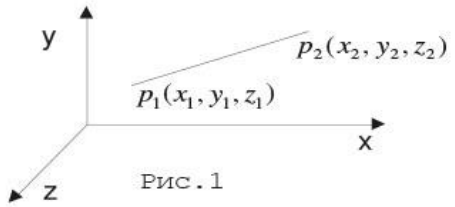


Рис.1

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2},$$

$$\alpha = \arccos\left(\frac{x_2 - x_1}{d}\right), \cdots \beta = \arccos\left(\frac{y_2 - y_1}{d}\right), \cdots \gamma = \arccos\left(\frac{z_2 - z_1}{d}\right).$$

x_1=25,5Λ; y_1=125Λ; z_1=0,56Λ;
x_2=75,5Λ; y_2=250Λ; z_2=125Λ;

Вывести значения d, α, β, γ.

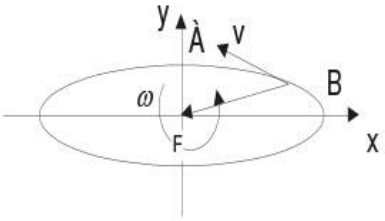
Задача №2.

Материальная точка массой m движется по эллипсу Рис. 1.

Составить программу для расчёта:

Координат точки в зависимости от времени в декартовой системе координат 0xyz по формулам:

$x(t) = A \cdot \cos\omega t$, $y(t) = B \cdot \sin\omega t$.



Проекции ускорения точки на оси ox и oy:

$ax = -\omega 2 \cdot A \cdot \cos\omega t$, $ay = -\omega 2 \cdot B \cdot \sin\omega t$.

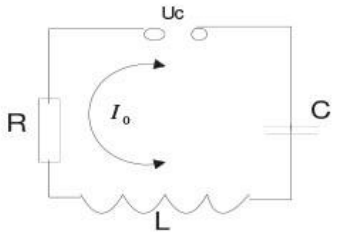
Проекции сил, действующих на материальную точку:

$Fx = -m \cdot \omega 2 \cdot A \cdot \cos\omega t$, $Fy = -m \cdot \omega 2 \cdot B \cdot \sin\omega t$.

Вывести значения вычисляемых величин, если известны

A=1000м, B=10000м, ω=5 $^{град}/_{сек}$, m=25г, t=50сек.

Задача №3.



Дан последовательный колебательный контур с параметрами элементов:

C=0,000003 Фарады, R=50 Ом, L=0.0101Генри.

К контуру приложено напряжение U0=12В, частота колебаний контура ω=2100Гц.

Составить программу для вычисления силы тока в контуре Io, падение напряжения на резисторе U0r, на конденсаторе U0c, на катушке индуктивности U0l и tgφ –тангенс угла φ между вектором тока и напряжения.

Формулы для вычислений:

$$U_0 = I_0 * R; \quad U_{0c} = \frac{I_0}{\omega * C}; \quad U_{0l} = I_0 * \omega * L;$$

$$I_0 = \frac{U_0}{\sqrt{R^2 + \left(\omega * L - \frac{1}{\omega * C}\right)^2}}; \quad \operatorname{tg}\varphi = \frac{\omega * L - \frac{1}{\omega * C}}{R};$$

При наличии достаточного времени можно запрограммировать вычисление I0 оператор – функцией.

4. Содержание отчета о практическом занятии

1. Титульный лист
2. Ответы на контрольные вопросы
3. Задачи
4. Решение
5. Вывод

Практическое занятие № 3-4

Программирование разветвляющихся алгоритмов

1. Цель занятия - получение практических навыков в решении задач на ЭВМ общего назначения, с использованием программ разветвляющихся алгоритмов.

2. Контрольные вопросы

1. Математическая формулировка задач.
2. Разработка разветвляющихся алгоритмов решения задач.
3. Программирование разветвляющихся алгоритмов.

3. Задание на практическое занятие

При подготовке к занятию необходимо изучить материалы лекций, и ответить на контрольные вопросы, используя рекомендованную литературу.

Цель занятия достигается путем получения навыков при решении представленных задач:

Задача № 1

Вычислить величину

$$y = \begin{cases} a + \dfrac{b}{x}, & \text{если } x \le b, \\ b - ax, & \text{если } x > b, \end{cases}$$

где $a = 3$;

$b = 45$;

$x = 31$

Вывести у в зависимости от аргумента х.

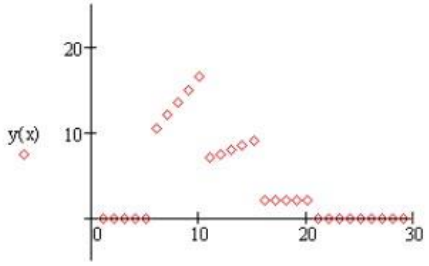
Задача № 2

Составить программу из задания № 1, используя команды условной и безусловной передачи управления, а также команду сравнению CMP.

Вывести y в зависимости от аргумента x.

Задача № 3

Имеется график функции Y(X), показанный на рисунке. Составить программу для вычисления значения функции Y(X) при любом значении аргумента X.



$$Y(X)=\begin{cases}0 & \text{если } 0<X\le 5,\\ 1.5\cdot(X+1) & \text{если } 5<X\le 10,\\ 1.5+0.5\cdot X & \text{если } 10<X\le 15,\\ 2.0 & \text{если } 15<X\le 20,\\ 0 & \text{если } X>20.\end{cases}$$

Задача № 4

Составить программу из задания № 3, используя команды условной и безусловной передачи управления, а также команду сравнению CMP.

Задача № 5.

Составить программу для вычисления значений величин q и α.

Вычисление запрограммировать при x=1, 3, 10;

$$q=\begin{cases}17x^2+0{,}75x, & \text{если } x\le 2,\\ 1{,}75x+\ln x, & \text{если } 2<x<7,\\ 4{,}2x^3+6{,}3x-4, & \text{если } x\ge 7.\end{cases}$$

$$\alpha=\begin{cases}\dfrac{q\cdot\sin^2 x}{0{,}7\cos x+0{,}3\sin x}, & \text{если } x<10,\\ \dfrac{q^2\cdot\sin x}{0{,}6\cos x+0{,}4\sin x}, & \text{если } 10\le x\le 20,\\ \dfrac{(1+q)\cdot\sin x}{0{,}5\cos x+0{,}5\sin x}, & \text{если } x>20.\end{cases}$$

Вывести q и α.

Задача № 6.

Составить программу для вычисления значений величин y и q:

$$y=\begin{cases}\dfrac{b}{a-x}, & \text{если } q<10,\\ \dfrac{a}{x-b}\cdot e^{-\frac{x^2}{a^2}}, & \text{если } q\ge 10.\end{cases}$$

$$q=\begin{cases}\sqrt{a-bx}, & \text{если } x<10,\\ \dfrac{e^{a\cdot b}}{a-bx}, & \text{если } 10\le x<20,\\ \dfrac{x}{a-x^2\cdot c^2}, & \text{если } 20\le x<30,\\ \sqrt[5]{x^2a^2+c^2}, & \text{если } x>40.\end{cases}$$

при следующих значениях переменных: a= -2.55, b= - 0.344, x=6, c=1.27; Вывести значения y и q вместе с именами.

4. Содержание отчета о практическом занятии

1. Титульный лист
2. Ответы на контрольные вопросы
3. Задачи
4. Решение
5. Вывод

Практическое занятие № 5

Программирование сложных циклических алгоритмов

1. Цель занятия - приобретение практических навыков по разработке сложных циклических алгоритмов и программ на алгоритмическом языке, включая алгоритмы обработки индексированных переменных.

2. Контрольные вопросы

1. Алгоритмы обработки индексированных переменных
2. Редактирование и отладка программ в ЭВМ
3. Реакций операционной системы на ошибки в программе

3. Задание на практическое занятие

При подготовке к занятию необходимо изучить материалы лекций, и ответить на контрольные вопросы, используя рекомендованную литературу.

Цель занятия достигается путем получения навыков при решении представленных задач:

Задача № 1

Вычислить переменную y, если аргументы a и x, от которых зависит y изменяются с постоянным шагом, так как показано в задании.

Запрограммировать вывод значений y при каждом значении a и x.

$$y = \begin{cases} a + x^2, & \text{если } x \geq a, \\ \dfrac{b}{a - x}, & \text{если } x < a, \end{cases}$$

где $b = 5 \cdot 10^{-2}$;

$a = 60, \ldots, 100$; $\Delta a = 20$;

$x = -20, \ldots, -200$; $\Delta x = -20$.

Задача № 2

Вычислить переменную y, если аргумент b, от которого зависит y изменяется с постоянным шагом, так как показано в задании, а аргумент x не имеет постоянного шага изменения, а изменяется произвольно.

Запрограммировать вывод значений y при каждом значении b и x.

$$y=\begin{cases} a+\dfrac{b}{x}, & \text{если } x \le b, \\ b-ax, & \text{если } x > b, \end{cases}$$

где $a=-3\cdot 10^{-2}$;

$b=0{,}45;\ \ldots;\ 0{,}75;\quad \Delta b=0{,}15$;

$x=0{,}31;\ 0{,}58;\ 0{,}93;\ 1{,}22;\ 1{,}5;\ 1{,}87$.

Задача № 3

Вычислить переменную y, если аргументы a и x, от которых зависит y, не изменяются с постоянным шагом, так как показано в задании.

Запрограммировать вывод значений y при каждом значении a и x. Ввод и вывод форматный.

$$y=\begin{cases} |ax-b|, & \text{если } a-bx \ge 0, \\ b+\dfrac{1}{ax}, & \text{если } a-bx < 0, \end{cases}$$

где $a=1{,}5$;

$b=4,-10,25,-17,8,12,32,-5,10,11,100.$

$x=1{,}52;3{,}05;4{,}5;6{,}2;7{,}4;8{,}3;9{,}1;11{,}7;12$.

4. Содержание отчета о практическом занятии

1. Титульный лист
2. Математическое описание решаемых задач;
3. Блок-схемы алгоритмов решаемых задач;
4. Таблицы идентификаторов;
5. Программы решения задачи в masm32;
6. Результаты решения задач (скриншоты).

Практическое занятие № 6-7

Программирование инициализации и ввода – вывода данных

1. Цель занятия - получение практических навыков в решении задач на ЭВМ общего назначения.

2. Контрольные вопросы

1. Математическая формулировка задач.
2. Разработка алгоритмов решения задач.
3. Программирование алгоритмов.

3. Задание на практическое занятие

При подготовке к занятию необходимо изучить материалы лекций, и ответить на контрольные вопросы, используя рекомендованную литературу.

Цель занятия достигается путем получения навыков при решении представленных задач:

Задача №1.

Составить программу для расчёта расстояния d между двумя точками и углов α, β, γ, наклона прямой к осям x,y,z .

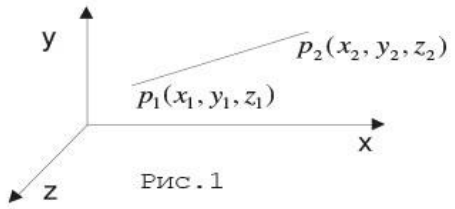


Рис . 1

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2},$$

$$\alpha = \arccos\left(\frac{x_2 - x_1}{d}\right), \cdots \beta = \arccos\left(\frac{y_2 - y_1}{d}\right), \cdots \gamma = \arccos\left(\frac{z_2 - z_1}{d}\right).$$

x_1=25,5Λ; y_1=125Λ; z_1=0,56Λ;
x_2=75,5Λ; y_2=250Λ; z_2=125Λ;

Вывести значения d, α, β, γ.

Предусмотреть ввод данных с клавиатуры, с использованием дочерних элементов управления операционной системы

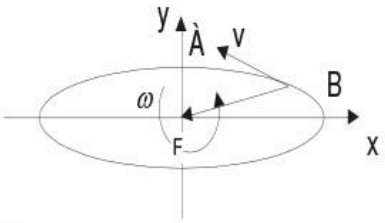
Задача №2.

Материальная точка массой m движется по эллипсу Рис. 1.

Составить программу для расчёта:

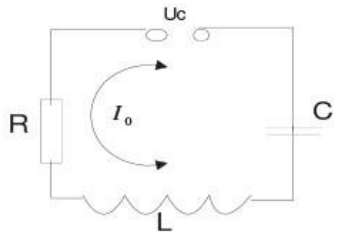
Координат точки в зависимости от времени в декартовой системе координат 0xyz по формулам:

x(t) = A·cosωt, y(t) = B·sinωt.



Проекции ускорения точки на оси ох и оу:
ax = -ω2·A·cosωt, ay = -ω2·B·sinωt.
Проекции сил, действующих на материальную точку:
Fx = -m·ω2·A·cosωt, Fy= -m·ω2·B·sinωt.
Вывести значения вычисляемых величин, если известны
A=1000м, B=10000м, ω=5 $^{\text{град}}/_{\text{сек}}$, m=25г, t=50сек.
<u>Предусмотреть ввод данных с клавиатуры, с использованием дочерних элементов управления операционной системы</u>

Задача №3.



Дан последовательный колебательный контур с параметрами элементов:
C=0,000003 Фарады, R=50 Ом, L=0.0101Генри.

К контуру приложено напряжение U0=12В, частота колебаний контура ω=2100Гц.

Составить программу для вычисления силы тока в контуре Io, падение напряжения на резисторе U0r, на конденсаторе U0c, на катушке индуктивности U0l и tgφ –тангенс угла φ между вектором тока и напряжения.

Формулы для вычислений:

$$U_0 = I_0 * R; \quad U_{0c} = \frac{I_0}{\omega * C}; \quad U_{0l} = I_0 * \omega * L;$$

$$I_0 = \frac{U_0}{\sqrt{R^2 + \left(\omega * L - \frac{1}{\omega * C}\right)^2}}; \quad \operatorname{tg}\varphi = \frac{\omega * L - \frac{1}{\omega * C}}{R};$$

При наличии достаточного времени можно запрограммировать вычисление I0 оператор – функцией.

Предусмотреть ввод данных с клавиатуры с использованием дочерних элементов управления операционной системы

4. Содержание отчета о практическом занятии

1. Титульный лист
2. Ответы на контрольные вопросы
3. Задачи
4. Решение
5. Вывод

Практическое занятие № 8

Обратная разработка. Базовый анализ исполняемых файлов

1. Цель занятия - получение базовых навыков анализа исполняемых файлов и использования инструментов для обратной разработки.

2. Контрольные вопросы

1. Инструменты для обратной разработки.
2. Работа с дизассемблером.
3. Метод анализа исполняемых файлов.

3. Теоретические сведения

Алгоритм дизассемблирования

Разработаем простой алгоритм принятия машинного языка в качестве входного и производства ассемблера в качестве выходного.

При этом мы получим представление о проблемах, предположениях и компромиссах, лежащих в основе автоматизированного процесса разборки.

1) Первым шагом в процессе разборки является определение секции кода для дизассемблирования. Это не обязательно так просто, как может показаться.

Инструкции, как правило, смешиваются с данными, и важно различать их. В наиболее распространенном случае, при дизассемблировании исполняемого файла, файл будет соответствовать общему для исполняемых файлов формату, такому как формат *Portable Executable (PE),* используемый в Windows, или *формат Executable and Linking Format (ELF),* распространенный во многих Unix-системах.

Эти форматы, как правило, содержат механизмы (часто в виде иерархических заголовков) для определения местоположения секций файла, содержащих код и точки входа в этот код.

Точка входа в программу - это просто адрес команды, на которую операционная система передает управление после загрузки программы в память.

2) Учитывая начальный адрес инструкции, следующим шагом является чтение значения, содержащегося по этому адресу (или смещение файла), и выполнение поиска таблицы для сопоставления значения двоичного опкода со значением его мнемонического языка ассемблера.

В зависимости от сложности дизассемблируемого набора инструкций, это может быть тривиальный процесс, или он может включать в себя несколько дополнительных операций, таких как понимание префиксов, которые могут изменить поведение инструкции, и определение операндов, требуемых инструкцией. Для наборов команд с инструкциями переменной длины, таких как Intel x86, может потребоваться извлечение дополнительных байтов команд для того, чтобы выполнить сложное дизассемблирование одной команды.

3) После того, как инструкция получена и все необходимые операнды декодированы, ее эквивалент на языке ассемблера форматируется и выводится как часть листинга разборки файла. Возможно выбрать синтаксис вывода на нескольких языках ассемблера.

4) После вывода команды нам нужно перейти к следующей команде и повторить предыдущий процесс до тех пор, пока мы не разберем каждую команду в файле.

Существуют различные алгоритмы определения того, с чего начинать дизассемблирование, как выбрать следующую разобранную инструкцию, как отличить код от данных и как определить, когда была разобрана последняя инструкция.

Два преобладающих алгоритма разборки файлов - это **линейная развертка и рекурсивный спуск.**

Линейная развертка

Алгоритм линейной зачистки при дизассемблировании использует очень простой подход к размещению инструкций по дизассемблированию: где одна инструкция заканчивается другая начинается. В результате, самое трудное решение - с чего начать. Обычное решение заключается в том, чтобы предположить, что все, что содержится в разделах программы, помеченных как код (обычно задается заголовками программных файлов), представляет собой инструкции машинного языка. Разборка начинается с первого байта в секции кода и линейно перемещается по секции, разбирая одну инструкцию за другой до тех пор, пока не будет достигнут конец секции. Не делается никаких усилий для понимания потока управления программой через распознавание нелинейных инструкций, таких как ветви.

Основным преимуществом алгоритма линейной развертки является то, что он обеспечивает полное покрытие участков кода программы.

Одним из основных недостатков метода линейной развертки является то, что он не учитывает тот факт, что данные могут смешиваться с кодом.

Линейная развертка используется механизмами дизассемблирования, содержащимися в отладчике GNU (gdb), отладчике Microsoft WinDbg и утилите objdump.

Рекурсивный спуск

При рекурсивном спуске используется другой подход к инструкциям по нахождению. Рекурсивный спуск фокусируется на концепции потока управления, которая определяет, должна ли инструкция быть разобрана или нет, исходя из того, ссылается ли она на другую инструкцию. Для понимания рекурсивного спуска полезно классифицировать инструкции в соответствии с тем, как они влияют на указатель на команду процессора.

Существуют:

- последовательные инструкции,
- условные инструкции по разветвлению,
- безусловные инструкции по разветвлению,
- инструкции по вызову функций и инструкции по возвращению.

Последовательные инструкции по потоку передают выполнение непосредственно следующей инструкции. Примеры последовательных инструкций потока включают в себя простые арифметические инструкции, такие как add; инструкции передачи регистров в память, такие как mov; и операции манипулирования стеком, такие как push и pop. Для таких инструкций разборка выполняется так же, как и при линейной зачистке.

Условные инструкции по разветвлению, такие как jnz, предлагают два возможных пути выполнения. Если условие вычисляется как истинное, то берется ветвь и указатель инструкции должен быть изменен, чтобы отразить цель ветви.

Однако, если условие ложное, выполнение продолжается линейным способом, и для разборки следующей инструкции можно использовать методологию линейной зачистки. Так как в статическом контексте обычно невозможно определить результат условного теста, алгоритм рекурсивного спуска разбирает оба пути, откладывая разборку целевой команды ветви, добавляя адрес целевой команды в список адресов, которые будут разобраны в последующем.

Нетрадиционные ветви не следуют линейной модели потока и поэтому по-разному обрабатываются алгоритмом рекурсивного спуска. Как и в случае с последовательными командами потока, выполнение может происходить только с одной командой; однако, эта команда не обязательно должна сразу же следовать за командой ветви.

Деассемблер рекурсивного спуска попытается определить цель безусловного перехода и добавить адрес назначения в список адресов, которые еще не изучены. К сожалению, некоторые безусловные ветви могут вызвать проблемы у рекурсивных дизассемблеров.

Когда цель команды перехода зависит от значения времени выполнения, определение адреса назначения jump с помощью статического анализа может оказаться невозможным.

Инструкции вызова функций работают очень похожим на безусловные инструкции jump (в том числе невозможность определить цель инструкций, таких как вызов регистра eax), с дополнительным ожиданием, что выполнение обычно возвращается к инструкции сразу после вызова, как только функция завершает работу. В этом отношении они схожи с условными инструкциями-ветвями в том, что генерируют два пути выполнения. Целевой адрес команды вызова добавляется в список для отложенной разборки, в то время как команда, следующая сразу за вызовом, разбирается аналогично линейной развертке.

Инструкция возврата функции (например, x86 ret) не дает никакой информации о том, какая инструкция будет выполняться дальше. Если бы программа действительно выполнялась, то адрес брался бы сверху стека времени выполнения, и выполнение возобновлялось бы по этому адресу. Дизассемблеры не имеют преимущества доступа к стеку. Вместо этого, разборка резко прекращается. Именно на этом этапе рекурсивного спуска дизассемблер обращается к списку адресов, которые он откладывал для отложенной разборки. Адрес удаляется из этого списка, и процесс разборки продолжается с этого адреса. Это рекурсивный процесс, который дает алгоритму разборки свое имя.

IDA Pro

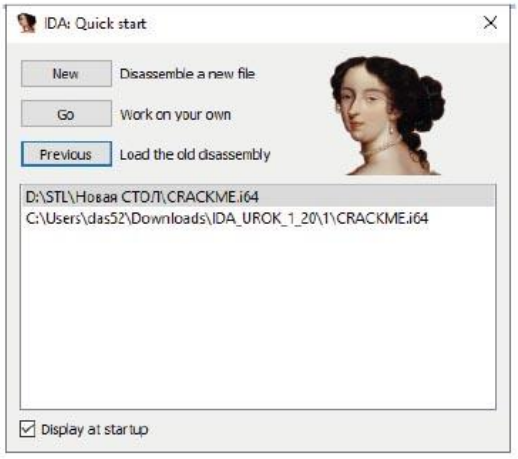
Interactive Disassembler Professional, лучший дизассемблер и до сих пор известный как IDA Pro, или просто IDA, является продуктом Hex-Rays. В течение многих лет Ида продавалась компанией DataRescue, однако с января 2008 года продается компанией Hex-Rays.

По своей сути IDA-это дизассемблер рекурсивного спуска, однако значительное количество усилий было потрачено на разработку логики для улучшения процесса рекурсивного спуска. Чтобы преодолеть один из самых больших недостатков рекурсивного спуска, IDA использует большое количество эвристических методов для идентификации дополнительного кода, который, возможно, не был найден в процессе рекурсивного спуска.

IDA прилагает все усилия, чтобы описать сгенерированные дизассемблированные файлы не только информацией о типе данных, но и производными именами переменных и функций. Эти описания минимизируют количество необработанного шестнадцатеричного кода и максимизируют количество символьной информации, представленной пользователю.

IDA содержит справочную документацию в стиле Windows, но это в первую очередь обзор пользовательского интерфейса IDA и подсистемы сценариев.

Установка IDA на Windows - это очень простой процесс. Запуск установщика Windows проведет вас через несколько информационных диалогов, только один из которых требует каких-либо размышлений:



Откуда можно скачать программу?

С официального сайта можно скачать IDA Pro Free (для некоммерческого использования) с рядом ограничений (версия, достаточная для первоначального знакомства с функционалом дизассемблера): http://www.hex-rays.com/idapro/idadownfreeware.htm

Благодаря плагинам, написанным на C++ функционал IDA Pro можно расширить. Обзор плагинов для IDA Pro (можно скачать без регистрации): http://www.openrce.org/downloads/browse/IDA_Plugins

Одним из плагинов для IDA Pro является декомпилятор HexRays, который переводит двоичный код в «C-подобный».

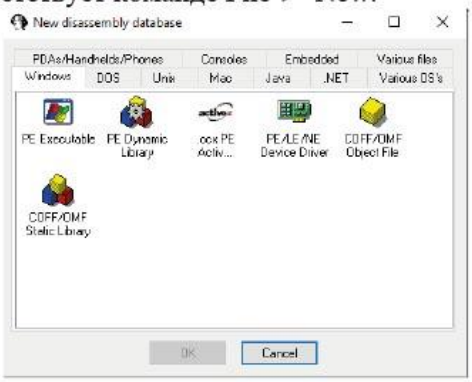
Каждый раз, когда вы запускаете IDA, вас коротко приветствует заставка, на которой отображается краткая информация о вашей лицензии.

NEW

Выбор пункта NEW запускает мастер, который проведет вас через весь процесс выбора нового файла для анализа. Как вы можете видеть в диалоговом окне, пользователи должны изначально определить тип файла, который они намерены открыть. Как только тип файла определен, для выбора файла, подлежащего анализу, используется стандартный диалог открытия файла. Наконец, отображается одно или несколько дополнительных диалоговых окон, которые позволяют выбрать определенные параметры анализа файлов перед загрузкой, анализом и отображением файла.

Одна из проблем с мастером создания новых файлов заключается в том, что пользователь должен точно знать, какой тип файла он открывает, что может быть или не быть так.

Кнопка New соответствует команде File ► New.



Go

Кнопка Go завершает процесс загрузки и вызывает открытие IDA с пустым рабочим пространством. На этом этапе, если вы хотите открыть файл, вы можете перетащить двоичный файл на рабочий стол IDA или использовать один из параметров меню Файл, чтобы открыть файл. Файл ► New command запускает мастер создания нового файла, как описано ранее.

Команда File ► Open заставляет IDA обойти Мастер создания нового файла и перейти прямо к диалоговому окну открытия файла. По умолчанию IDA использует известный фильтр расширений для ограничения представления диалогового окна файла. Когда вы открываете файл таким образом, IDA пытается автоматически определить тип выбранного файла; однако вы должны внимательно следить за диалогом загрузки, чтобы увидеть, какие загрузчики были выбраны для обработки файла.

При выборе открытия нового файла с помощью команды File ► Open вам будет представлен диалог загрузки, показанный на ниже рисунке.

IDA генерирует список потенциальных типов файлов и отображает этот список в верхней части диалогового окна. В этом списке представлены загрузчики IDA, которые лучше всего подходят для работы с выбранным файлом.

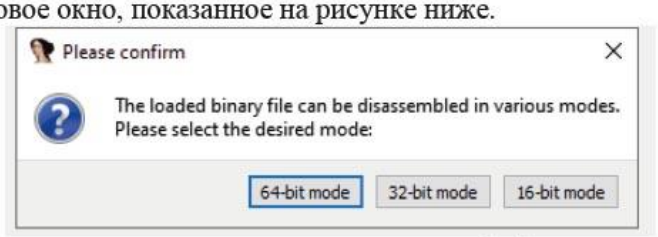
Иногда двоичный файл будет единственной записью, которая появится в списке загрузчиков.

В таких случаях подразумевается, что ни один из загрузчиков не распознает выбранный файл. Если вы решили продолжить процесс загрузки, убедитесь, что вы выбрали тип процессора в соответствии с вашим пониманием содержимого файла.

Использование загрузчика двоичных файлов

Когда вы решите использовать двоичный загрузчик, вы должны быть готовы сделать больше, чем вы обычно делали. При отсутствии информации о заголовке файла, вы можете вмешаться в процесс анализа и выполнить задачи, которые более способные загрузчики часто выполняют автоматически.

Примеры ситуаций, которые могут потребовать использования двоичного загрузчика, включают анализ образов ПЗУ и полезных нагрузок эксплойтов, которые могут быть извлечены из сетевых пакетов захвата или файлов журнала. Когда процессорный модуль x86 будет сопряжен с двоичным загрузчиком, появится диалоговое окно, показанное на рисунке ниже.



При отсутствии распознаваемых заголовков файлов, доступных для помощи IDA, пользователь должен указать, следует ли рассматривать код как 16-разрядный или 32-разрядный код режима. Другие процессоры, для которых IDA может различать 16-и 32-разрядные режимы, включают ARM и MIPS.

Двоичные файлы не содержат никакой информации, касающейся их расположения в памяти (то есть, по крайней мере, никакой информации, которую IDA умеет распознавать. Если выбран тип процессора x86, то информация о базовом адресе должна быть указана в полях сегмент загрузки и смещение загрузки диалогового окна загрузчика, как упоминалось ранее.

В IDA нет никакой отмены.

Если что-то непредвиденное произойдет с вашей базой данных в результате непреднамеренного нажатия клавиши, вы сами сможете восстановить свои дисплеи в прежнее состояние. Почти все действия имеют соответствующий пункт меню, горячую клавишу и кнопку панели инструментов. Помните, что панель инструментов IDA легко настраивается, как и отображение горячих клавиш действий меню. IDA предлагает хорошие, контекстно-зависимые действия меню в ответ на щелчки правой кнопкой

Основные дисплеи IDA

В своей конфигурации по умолчанию IDA создает семь (начиная с версии 6.1) отображаемых окон во время начальной фазы загрузки и анализа для нового двоичного файла. Каждое из этих окон отображения доступно через

набор вкладок заголовка. Три непосредственно видимых окна - это окно IDA-View, окно функций и окно вывода. Независимо от того, открыты ли они по умолчанию, все окна, описанные в этой главе, можно открыть с помощью меню View ► Open Subviews. Имейте это в виду, так как довольно легко непреднамеренно закрыть окна дисплея.

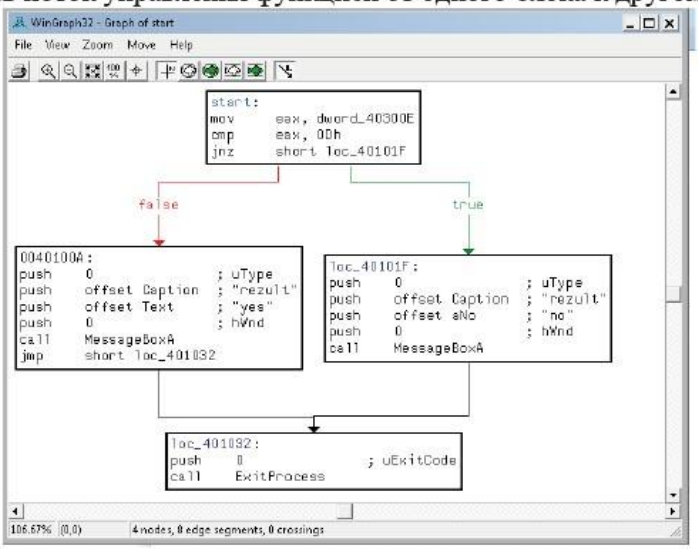
Клавиша ESC - одна из самых полезных горячих клавиш во всей IDA. Когда окно демонтажа активировано, клавиша ESC функционирует аналогично кнопке "Назад" веб-браузера и поэтому очень полезна для навигации по дисплею демонтажа.

Окно Дизассемблирования

Также известное как окно IDA-View, окно дизассемблирования является основным инструментом для манипулирования и анализа двоичных файлов. Для окна дизассемблирования доступны два формата отображения: представление на основе графики по умолчанию и текстовое представление листинга. Большинство пользователей IDA склонны предпочесть одно представление другому, и представление, которое лучше соответствует вашим потребностям, часто определяется тем, как вы предпочитаете визуализировать поток программы.

IDA Graph View

На рисунке показана простая функция, отображаемая в виде графика. Графические представления несколько напоминают программные блок-схемы в том смысле, что функция разбивается на базовые блоки, так что вы можете визуализировать поток управления функцией от одного блока к другому.



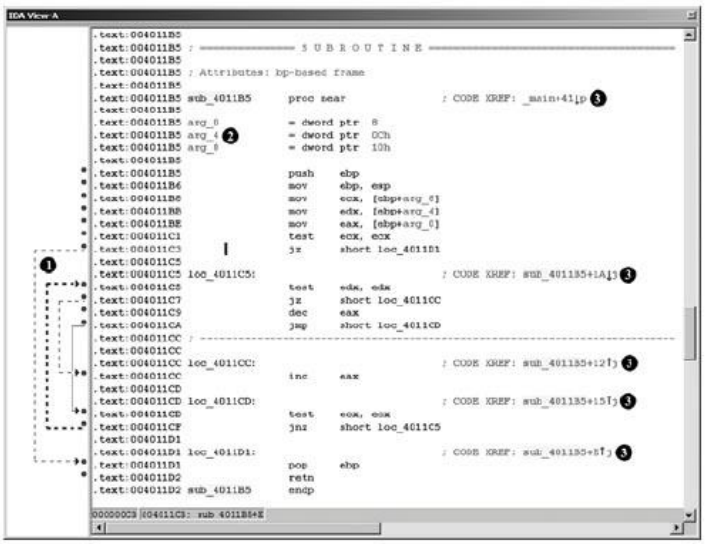
На экране вы заметите, что IDA использует различные цветные стрелки, чтобы различать различные типы потоков между блоками функции.

Базовые блоки, которые заканчиваются только одним потенциальным блоком-преемником, используют обычное ребро (по умолчанию синее), чтобы указать на следующий блок, который будет выполняться.

В графическом режиме IDA отображает одну функцию за раз.

Большие или сложные функции могут привести к тому, что представление графика будет чрезвычайно загромождено, что затруднит навигацию по нему. В окне обзора всегда отображается полная блочная структура графики, а также пунктирная рамка, указывающая область графики, которая в данный момент просматривается в окне disasembly. Пунктирную рамку можно перетащить через обзорное окно, чтобы быстро переместить вид графика в любое желаемое место на графике.

Посмотреть текст IDA



Текстовое окно дизассемблирования - это традиционный дисплей, используемый для просмотра и манипулирования созданными IDA. Текстовый дисплей представляет весь список демонтажа программы (в отличие от одной функции за один раз в графическом режиме) и обеспечивает единственное средство для просмотра областей данных двоичного файла. Вся информация, доступная в графическом отображении, доступна в текстовом отображении в той или иной форме.

Дизассемблирование представлено линейным способом, с виртуальными адресами, отображаемыми по умолчанию.

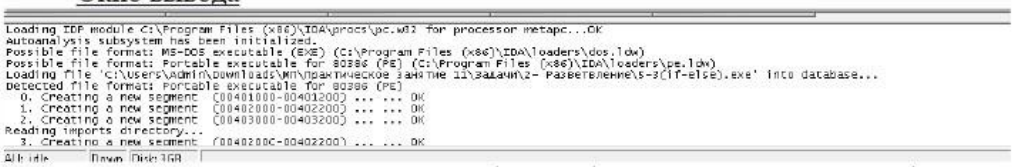
Окно Функций

Окно функции используется для перечисления всех функций, которые IDA распознала в базе данных. Запись в окне функций может выглядеть следующим образом:

malloc	.text	00BDC260 00000180 R . . . B . .

Эта конкретная строка указывает на то, что функция malloc может быть найдена в поле .text раздел двоичного файла с виртуальным адресом 00BDC260 имеет длину 384 байта (hex 180), возвращается вызывающему объекту (R) и использует регистр EBP (B) для ссылки на его локальные переменные. Флаги, используемые для описания функции (например, R и B выше), описываются во встроенном файле справки IDA (или при щелчке правой кнопкой мыши на функции и выборе свойства). Флаги отображаются в виде редактируемых флажков в результирующем диалоговом окне свойств). Как и в других окнах отображения, двойной щелчок по записи в окне функции приводит к тому, что окно разборки переходит в расположение выбранной функции.

Окно вывода



Окно вывода в нижней части рабочей области IDA завершает набор окон по умолчанию, которые отображаются при открытии нового файла. Окно Ouput служит консолью вывода IDA и является местом для поиска информации о задачах, которые выполняет IDA. Например, при первом открытии двоичного файла генерируются сообщения, указывающие как на то, на каком этапе анализа находится IDA в данный момент времени, так и на то, какие действия IDA выполняет для создания новой базы данных. При работе с базой данных окно вывода используется для вывода состояния различных выполняемых операций. Содержимое окна вывода можно скопировать в системный буфер обмена или полностью очистить, щелкнув правой кнопкой мыши в любом месте окна и выбрав соответствующую операцию. Окно вывода часто является основным средством отображения выходных данных из любых сценариев и подключаемых модулей, разрабатываемых для IDA.

Вторичные IDA дисплеи

Помимо окон разборки, функций и вывода, IDA открывает ряд других окон с вкладками на рабочем столе IDA. Эти вкладки находятся непосредственно под полосой навигации. Эти окна используются для предоставления альтернативных или специализированных представлений в базу данных. Полезность этих окон зависит как от характеристик двоичного файла, который вы анализируете, так и от ваших навыков работы с IDA.

На первый взгляд, количество окон, которые предлагает IDA, может показаться подавляющим. Вы можете решить, что проще всего придерживаться основных дисплеев, пока вы не почувствуете себя достаточно комфортно, чтобы начать изучать дополнительные предложения дисплея. В любом случае, вы не должны чувствовать себя обязанным использовать все, что IDA предоставляет вам. Не каждое окно будет полезно в каждом сценарии обратного проектирования.

4. Задание на практическое занятие

При подготовке к занятию необходимо изучить материалы лекций, и ответить на контрольные вопросы, используя рекомендованную литературу.

Цель занятия достигается путем выполнения списка заданий:

1. Изучить теоретические сведения.
2. Получить задания у преподавателя.
3. Произвести обратную разработку 3 исполняемых файлов, в первом и втором исполняемых файлах найти контрольные значения.

В третьем исполняемом файле найти число (или числа, если их несколько), при вводе которого (которых) происходит выдача сообщения о верности секретного ключа.

4. Составить отчёт. Стиль оформления отчёта – научно-технический.

5. Содержание отчета о практическом занятии

1. Титульный лист.
2. Описание по каждому пункту (со скриншотами).
3. Для последнего задания должна быть приведена формула, по которой происходит преобразование введённого значения, и обоснована верность полученного ответа
3. Вывод.