****

**ПРЕДМЕТ:**

**“Стили программирования и парадигмы”**

**САМОСТОЯТЕЛЬНАЯ РАБОТА**

**Тема:** Языки макро обработки текстов

**Выполнил: студент Абдурахмонов Самандар группы 655-20**

**«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_ г.**

**Принял(-а): преподаватель. Р.Зулунов**

**Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_\_\_ г.**

**Аннотация:**В данной самостоятельной работе рассматривается понятие языков макрообработки текста. Изучается устройство ряда макропроцессоров, используемых при обеспечении гибкости кода программ, рассматриваются макропроцессоры нестандартных языков программирования, применявшиеся при факторизации текстов программ, разрабатываемых одновременно на разные архитектуры. Приведены примеры кода программ на языках макрообработки текста

**Ключевые слова:**[GPM](https://intuit.ru/studies/professional_retraining/941/courses/204/lecture/5283?page=1" \l "keyword1), [информационные системы](https://intuit.ru/studies/professional_retraining/941/courses/204/lecture/5283?page=1" \l "keyword2), [класс](https://intuit.ru/studies/professional_retraining/941/courses/204/lecture/5283?page=1" \l "keyword3), [глобальные переменные](https://intuit.ru/studies/professional_retraining/941/courses/204/lecture/5283?page=1" \l "keyword4), [компилятор](https://intuit.ru/studies/professional_retraining/941/courses/204/lecture/5283?page=1" \l "keyword5), [текстовый редактор](https://intuit.ru/studies/professional_retraining/941/courses/204/lecture/5283?page=1" \l "keyword6), [макроопределения](https://intuit.ru/studies/professional_retraining/941/courses/204/lecture/5283?page=1" \l "keyword7), [синтаксическое подобие](https://intuit.ru/studies/professional_retraining/941/courses/204/lecture/5283?page=1" \l "keyword8), [механизмы](https://intuit.ru/studies/professional_retraining/941/courses/204/lecture/5283?page=1" \l "keyword9), [snobol](https://intuit.ru/studies/professional_retraining/941/courses/204/lecture/5283?page=1" \l "keyword10), [макрос](https://intuit.ru/studies/professional_retraining/941/courses/204/lecture/5283?page=1" \l "keyword12), [определение](https://intuit.ru/studies/professional_retraining/941/courses/204/lecture/5283?page=1" \l "keyword13), [кодогенерация](https://intuit.ru/studies/professional_retraining/941/courses/204/lecture/5283?page=1" \l "keyword14), [выделение подстроки](https://intuit.ru/studies/professional_retraining/941/courses/204/lecture/5283?page=1" \l "keyword15), [минимизация](https://intuit.ru/studies/professional_retraining/941/courses/204/lecture/5283?page=1" \l "keyword17), [константы](https://intuit.ru/studies/professional_retraining/941/courses/204/lecture/5283?page=1" \l "keyword18), [входной](https://intuit.ru/studies/professional_retraining/941/courses/204/lecture/5283?page=1" \l "keyword19), [универсальная функция](https://intuit.ru/studies/professional_retraining/941/courses/204/lecture/5283?page=2" \l "keyword22)

Опе­ра­тор, ра­бо­таю­щий в ка­кой-ли­бо сис­те­ме, час­то встре­ча­ет­ся с не­об­хо­ди­мо­стью по­вто­рять не­ко­то­рые по­сле­до­ва­тель­но­сти дей­ст­вий мно­го раз? Та­кая по­сле­до­ва­тель­ность мо­жет, на­при­мер, со­сто­ять из вво­да не­ко­то­рой тек­сто­вой по­сле­до­ва­тель­но­сти, на­жа­тии оп­ре­де­лен­ной по­сле­до­ва­тель­но­сти кла­виш, вы­пол­не­нии од­но­тип­но­го ря­да ка­ких-ли­бо ариф­ме­ти­че­ских опе­ра­ций. В по­доб­ных слу­ча­ях час­то мож­но вос­поль­зо­вать­ся ап­па­ра­том мак­ро­ко­манд. Мак­ро­ко­ман­ды (час­то на­зы­вае­мые мак­ро или мак­рос) яв­ля­ют­ся од­но­строч­ны­ми со­кра­ще­ния­ми для груп­пы ко­манд. Ис­поль­зуя мак­ро­ко­ман­ду, про­грам­мист по су­ще­ст­ву оп­ре­де­ля­ет од­ну “ко­ман­ду” для пред­став­ле­ния не­ко­то­рой по­сле­до­ва­тель­но­сти ко­манд. Оп­ре­де­ляя со­от­вет­ст­вую­щие мак­ро­ко­ман­ды, опе­ра­тор мо­жет удоб­ным для се­бя об­ра­зом вво­дить свои соб­ст­вен­ные сред­ст­ва бо­лее вы­со­ко­го уров­ня, не за­бо­тясь о струк­ту­ре сис­те­мы. Он мо­жет дос­тиг­нуть крат­ко­сти и про­сто­ты управ­ле­ния сис­те­мой, не те­ряя при этом ос­нов­ных преимуществ ис­поль­зо­ва­ния ис­ход­ной сис­те­мы, та­кой, как на­при­мер язык ас­семб­ле­ра. Круп­ные мак­ро­опе­ра­ции уп­ро­ща­ют поль­зо­ва­ние, от­лад­ку и мо­ди­фи­ка­цию про­грамм, и об­лег­ча­ют стан­дар­ти­за­цию. Мно­гие раз­ра­бот­чик вы­чис­ли­тель­ных ма­шин ис­поль­зу­ют мак­ро­ко­ман­ды для ав­то­ма­ти­за­ции со­став­ле­ния “под­хо­дя­щих” опе­ра­ци­он­ных сис­тем в про­цес­се, на­зы­вае­мом ге­не­ра­ци­ей сис­те­мы

Изучается устройство ряда макропроцессоров, используемых при обеспечении гибкости кода программ. Кроме достаточно известных *GPM* и Trac, ассемблерной макротехники и препроцессоров систем программирования, рассматриваются макропроцессоры нестандартных языков программирования, применявшиеся при факторизации текстов программ, разрабатываемых одновременно на разные архитектуры.

Макропроцессоры - мощный инструмент повышения емкости действий, образующих процессы информационной обработки. Главное предназначение макросов в системах программирования - достижение гибкости и переносимости текстов программ, применяемых в разных условиях. Многие трудности такого применения макротехники связаны с проблемой контроля типов данных на уровне текста программы. Исследования систем переписывания термов и разметки текстов пока не дали практичных решений в этой области. Современные *информационные системы* как правило содержат макропроцессоры как инструмент настройки на различные стандарты подготовки и обработки данных.

Начнем с классификации макропреобразований по мощности допустимых воздействий на обрабатываемые данные, предложенной в 1973 году А.А. Берсом:

* чисто текстовая подстановка, обеспечивающая подготовку текстов по шаблонам и формулярам;
* условная подстановка, допускающая при формировании текста использование статических параметров;
* вычисляемые макропреобразования - вычисляются значения некоторых формул, возможно с использованием средств основного языка программирования.

Любой *класс* макропреобразований может использовать локальные или *глобальные переменные*, вложенность областей действия определений, рекурсию. Макропроцессор может быть встроен в *компилятор*, быть автономным инструментом системы программирования, таким как *текстовый редактор*, оптимизатор или отладчик, или существовать самостоятельно как универсальный инструмент общего назначения.

Обычно *макроопределения* формируются с учетом границ строк, макровызовы могут выполняться за один просмотр или до исчерпания при итеративном анализе текста. Возможно управление глубиной макроподстановки. Популярно *синтаксическое подобие* макросов выражениям базового языка, хотя это может вызывать путаницу в понимании реальных механизмов при порождении кода программы.

Псев­до­ко­ман­да MACRO - пер­вая стро­ка оп­ре­де­ле­ния - оп­ре­де­ля­ет сле­дую­щий за ней иден­ти­фи­ка­тор, как имя мак­ро­ко­ман­ды. Вслед за этой стро­кой рас­по­ла­га­ет­ся по­сле­до­ва­тель­ность ко­манд, на­зы­вае­мых “те­лом мак­ро­оп­ре­де­ле­ния”.  Оп­ре­де­ле­ние за­кан­чи­ва­ет­ся стро­кой с псев­до­ко­ман­дой END. Ес­ли мак­ро­ко­ман­да оп­ре­де­ле­на, то ис­поль­зо­ва­ние име­ни со­от­вет­ст­вую­щей мак­ро­ко­ман­ды в ка­че­ст­ве мне­мо­ни­ки ко­да в про­грам­ме эк­ви­ва­лент­но ис­поль­зо­ва­нию со­от­вет­ст­вую­щей по­сле­до­ва­тель­но­сти ко­манд.

Для автономных макропроцессоров характерны специальные *механизмы* регулярного конструирования различимых текстов:

* управляющие символы;
* счетчики;
* генераторы уникальных значений;
* блоки активности (разметка);
* встроенные функции;
* управление вводом-выводом.

Макротехника обладает родством с методами конструирования регулярных выражений, техникой сопоставления данных с образцом (*Snobol*, Prolog), системами переписывания и современными языками разметки (xml, TeX и др.).

Макросом называют средство замены строки на другую, полученную из исходной по заранее заданным правилам. Такую замену осуществляет макропроцесор, управляемый системой макросов. Макропроцессор перерабатывает текст, содержащий вызовы макроса и новые *макроопределения*, пополняющие систему макросов. Различают общие и локальные макросы, воздействующие на всю текущую программу или на часть ее текста. Системы макросов могут организовываться в библиотеки.

Общеизвестно, что *макрос* легче применять, чем определять. Внешняя простота введения макросов сопряжена с вероятностью трудно обнаруживаемых ошибок периода исполнения программы, индуцированных случайным сходством с подпрограммами на основном языке программирования:

* макрос меняет текст программы,
* подпрограмма меняет данные программы и логику процесса исполнения программы.

Достаточно распространено *определение* макросов для нужд одной единственной программы, образующих с ней единое целое.

Большинство макропроцессоров допускают переменные макропериода - глобальные и локальные макропеременные.

Макротехника используется при автоматизации формирования различимых имен, например, различимые меток или имен переменных в программе.

Встречается интересное применение вложенности макровызовов и макроопределений, включая рекурсию.

ряд = | <сч = 0> -> [ 0 ]

| |

| [ ( | сч | + 1]

| |

| ряд [ сч - 1 ]

| |

| [ ) ]

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ряд (6) = (6+ (5+ (4+ (3+ (2+ (1+0))))))

Пример 1.

Макротехника результативна не только на текстах, но и на геометрических фигурах, графах и кодах. Например, макросами можно описать всем известное пентамино, оптимизацию и *кодогенерацию* программ.

Техника строковой обработки обычно поддерживается операциями вычисления длины строки, *выделения подстроки* и конкатенации строк.

При поддержке динамически возникающих макроопределений традиционно обеспечивают самодостаточность, т.е. возможность развития макропроцессора своими средствами. Обычные требования:

* Подстановка аргументов
* Использование библиотек
* Допущение переменных и структур данных
* Присвоение значений переменных
* Ветвления и переходы
* Циклы, иерархия и рекурсия
* Динамика обработки и создания новых макросов

Макропроцессор получает логическое завершение, поддерживая динамически формируемые *макроопределения*, возможно используя макровызовы в аргументах. Являясь инструментом расширения средств системы программирования, макропроцессор по своей природе должен быть развиваемым. Традиционная при разработке систем программирования общность и *минимизация* понятий оказывается не лучшим идеалом для разработки информационных систем массового назначения потому, что решение практических задач всегда сопряжено со множеством тривиальных мелочей, связанных с нелогичными особенностями оборудования или привычками пользователей. Макротехника дает сравнительно недорогой метод учета таких мелочей на уровне специализации системных приложений.

На практике макроассемблер выполняет роль расширенной системы команд. Такие команды могут обеспечивать специальные модули для обработки файлов с нестандартной информацией, например, распознавать текст с устаревшими или нестандартными шрифтами.

Отдельного решения требуют вопросы проявления в программах контекстов без макрообработки, таких как строковые *константы* и комментарии, выпадающие из общего строя языка программирования.

В системах программирования препроцессоры обычно формируют *входной* текст для компилятора, а макроассемблеры выполняют сборку кода на уровне ассемблера или объектного кода.

В текстовых процессорах - контекстная замена, контекстное редактирование и регулярные преобразования текста.

Техника выполнения макропреобразований достаточно разнообразна. Так, например, язык *GPM* всю работу с макросами сводит к макровызову вида:

$ mak, a1, a2, ... aN; - вызов макроса1

Позиции макровызова занумерованы по числу предшествующих запятых, что делает не нужным описание переменных, одновременно с возможностью самоприменения определений.

~0 ~1 ~2 ... ~N - описание не нужно

Кроме того используются скобки, блокирующие подстановки при необходимости.

< S > - блокировка подстановок в S

Достаточно всего одной встроенной функции DEF, выполняющей введение макропределений.

$ DEF, mak, опр;

Пример 2. Введение новых макроопределений GPM

$ Def, size, 6;

$ size; => 6

x ($size, $size) => x(6,6)

size$size => size6

Пример 3. Использование макроопределений GPM

$Def, opp, UN~1;

$opp, R; => ОШ - нет аргумента

$Def, opp, <UN~1>;

$opp, R; => UNR

Пример 4. Использование блокировок в макроопределениях GPM

if A=B then C else D

$A, $Def, A , <D>; $def, B, <C>;;

Пример 5 Моделирование ветвлений макроопределенем GPM

В этом примере результат условного выражения обеспечен побочным эффектом на глобальной таблице имен.

Совершенно иначе выглядит макротехника в не менее лаконичном языке макропроцессора TRAC. Все сводится к макровызовам функций, встроенных и определяемых.

# (F, s1,s2,...,sN)

Встроенные функции:

ds - опр. строки

cl - call

ss - выделить сегмент

rs - чтение строки

#(DS,ПРИМЕР, собака сидит на ковре)

#(ss,ПРИМЕР,собака,ковре)

#(cl,ПРИМЕР,кошка,кресле) = кошка сидит на кресле

Пример 6. Работа с шаблонами на языке Trac

Два интересных механизма макротехники были реализованы в проекте языка Setl при попытке его эффективной реализации посредством языка Little [[[ 49 ]](https://intuit.ru/studies/professional_retraining/941/courses/204/literature" \l "literature.49),[[ 32 ]](https://intuit.ru/studies/professional_retraining/941/courses/204/literature" \l "literature.32)].

Для поддержки переноса программы на разные архитектуры предлагался специальная разметка текста с помощью флагов, в зависимости от значения которых блоки строк включались во *входной* текст для компилятора. Значения флагов можно было инициировать, наращивать или редуцировать и обнулять.

ОПЕ­РА­ТО­РЫ МАК­РО­КО­МАНД Ап­па­рат мак­ро­ко­манд в том ви­де, как он был опи­сан до сих пор, по­зво­ля­ет под­став­лять по­сле­до­ва­тель­но­сти ко­манд вме­сто мак­ро­вы­зо­вов, при­чем все об­ра­ще­ния к мак­ро­оп­ре­де­ле­нию бу­дут за­ме­не­ны иден­тич­ны­ми по­сле­до­ва­тель­но­стя­ми ко­манд. Та­кой ап­па­рат не­дос­та­точ­но ги­бок: в мак­ро­вы­зо­ве нет средств мо­ди­фи­ци­ро­вать ко­ды, ко­то­рые его за­ме­ня­ют. Су­ще­ст­вен­ное рас­ши­ре­ние воз­мож­но­стей мак­ро­средств дос­ти­га­ет­ся до­бав­ле­ни­ем опе­ран­дов (па­ра­мет­ров) мак­ро­ко­манд.

+ flag - включить строку.

.flag - завершение блока, сопровождается увеличением или

уменьшением счетчика, одноименного с флагом.

- flag - пропустить строку.

Для автоматизации формирования фрагментов текста, обладающих зависимостью от численных характеристик или кратности вхождения в программу использовался специальный механизм специальных макропеременных:

zxN => N + I --- в строке размещается значение счетчика

zyN = N' => N' (zyN := N') --- задание значения спецпеременной

zaN => A(N+i) --- в строке размещается имя "а",

сцепленное со значением счетчика

Пример 7. Представление зависимости от процесса формирования текста

В общем случае интерпретирующие автоматы для макропроцессоров характеризуются возможностью многократной обработки данных, до тех пор пока не будет получен результат без макровызовов. Соответствующая *универсальная функция* построения результата макрообработки получается реализационно не очень простой.

Подобные *механизмы* макрообработки текстов используются препроцессорами в стандартных системах программирования и текстовых процессорах. Иногда встречаются и более специализированные средства, использующие счетчиковые переменные, конструкторы уникальных имен, моделирующие иерархию модулей или параметризующие зависимость вариантов программы от целевых архитектур.

Концептуально макротехника близка продукционному стилю программирования, языкам разметки и системам переписывания текстов, в настоящее время активно развивающимся как языки гипертекстов для разработки сайтов и информационных сервисов.

**Заключение**

Макроязыки и соответствующие им макропроцессоры представляют собой самостоятельную форму языков программирования. При использовании вместе с ассемблером, макропроцессор является для программиста полезным инструментом и по существу, позволяет ему самому определять свой язык “вы­со­ко­го” уровня. Существуют четыре основных задачи, решаемых макропроцессором: 1. Распознавание макроопределений 2. Хранение макроопределений 3. Распознавание макрокоманд 4. Расширение макрокоманд и подстановка параметров Макропроцессор в ассемблере может быть реализован несколькими способами: 1. Независимый двухпросмотровый ассемблер 2. Независимый однопросмотровый ассемблер 3. Процессор, совмещенный с первым проходом стандартного двухпросмотрового ассемблера.

**Литература**:

1. Дж. Джор­дан - “Сис­тем­ное про­грам­ми­ро­ва­ние”
2. IBM OS/2 - “REXX Programmer’s Reference”
3. Borland C++ - Included documentation and sources.
4. <https://intuit.ru/studies/professional_retraining/941/courses/204>
5. <https://www.bibliofond.ru/view.aspx?id=2802>