***Тема 6. Программное обеспечение и технологии программирования***

**1. Системы программирования**

***Система программирования*** – это комплекс средств, предназначенный для создания и эксплуатации программ на конкретном языке программирования на ЭВМ определенного типа.

**Средства для создания программ**

Традиционными средствами разработки программ являются алгоритмические (процедурные) языки программирования. Для создания программы на выбранном языке программирования нужно иметь следующие компоненты:

***Текстовый редактор*** – это редактор, который позволяет набрать текст программы на языке программирования. Для этой цели можно использовать любые текстовые редакторы, но лучше пользоваться специализированным текстовым редактором.

***Транслятор*** – это основа систем программирования*.* Трансляторы языков программирования, т. е. программы, обеспечивающие перевод исходного текста программы на машинный язык (объектный код), бывают двух типов: интерпретаторы и компиляторы.

*Интерпретатор* - это транслятор, который обеспечивает последовательный синхронный «перевод» и выполнение каждой строки программы, причем при каждом запуске программы на выполнение вся процедура полностью повторяется. Достоинством интерпретатора является удобство отладки программы в интерактивном режиме, а недостатком - малая скорость работы.

*Компилятор* – это транслятор, который исходный текст программы переводит в *машинный код*. Если в тексте программы нет синтаксических ошибок, то машинный код будет создан. Но это, как правило, не работоспособный код, т.к. в этой программе не хватает подпрограмм стандартных функций, поэтому компилятор выдает промежуточный код, который называется *объектным кодом* и имеет расширение ***.obj***.

***Редактор связей (сборщик)*** – это программа, которая объединяет объектные модули отдельных частей программы и добавляет к ним стандартные модули подпрограмм стандартных функций (файлы с расширением ***.lib***), которые содержатся в библиотеках, поставляемых вместе с компилятором, в единую программу, готовую к исполнению, т.е. создает *исполнимый* ***.exe*** файл. Этот файл имеет самостоятельное значение и может работать под управлением той (или такой же) операционной системы, в которой он создан.

**Интегрированные системы программирования**

В стандартную поставку, как правило, входят текстовый редактор, компилятор, редактор связей (сборщик), библиотеки стандартных функций. Но хорошая *интегрированная система* обязательно включает в себя специализированный текстовый редактор, в котором выделяются ключевые слова различными цветами и шрифтами. Все этапы создания программы в ней автоматизированы: после того, как исходный текст программы введен, его компиляция и сборка осуществляются одним нажатием клавиши.

В современныхинтегрированных системах имеется еще один компонент – ***отладчик.*** Он позволяет анализировать работу программы по шагам во время ее выполнения, наблюдая, как меняются значения различных переменных.

В последние несколько лет созданы *среды быстрого проектирования*, в которых программирование, по сути, заменяется проектированием. В проектируемое окно готовые визуальные компоненты перетаскиваются с помощью мыши, затем свойства и поведение компонентов настраивается с помощью редактора. Исходный же текст программы, ответственный за работу этих элементов, генерируется автоматически с помощью *среды быстрого проектирования*, которая называется ***RAD***-средой. Подобный подход называется *визуальным программированием*.

**Основные системы программирования**

|  |  |
| --- | --- |
| **Наиболее популярные языки программирования** | **Соответствующие им визуальные среды быстрого проектирования программ для Windows** |
| ***Бейсик (Basic)*** – для освоения требует начальной подготовки (общеобразовательная школа) | ***Microsoft Visual Basic*** |
| ***Паскаль (Pascal)*** – требует специальной подготовки | ***Borland Delphi*** |
| ***Си++ (С++)-*** требует серьезной подготовки | ***Microsoft Visual С++*** |
| ***Ява (Java)*** требует серьезной подготовки | ***Java: Borland JBuilder*** |

***2. Структурное программирование***

Основная идея структурного программирования состоит в том, что структура программы должна отражать структуру решаемой задачи, чтобы алгоритм программы был ясно виден из исходного текста. Следовательно, надо разбить программу на последовательность модулей, каждый из которых выполняет одно или несколько действий. Требование к модулю – чтобы его выполнение начиналось с первой команды и заканчивалось последней. Модульность – это основная характеристика структурного программирования. А для этого надо иметь средства для создания программы не только с помощью трех простых операторов, но и с помощью средств более точно отражающих конкретную структуру алгоритма.

**С этой целью в программирование введено понятие *подпрограммы* – набора операторов, выполняющих нужное действие и не зависящих от других частей исходного кода.** Программа разбивается на множество подпрограмм, каждая из которых выполняет одно из действий исходного кода. Комбинируя эти блоки, удается сформировать итоговый алгоритм уже не из операторов, а из законченных блоков. Обращаться к блокам надо по названиям, а название несет смысловую нагрузку. Например, Call Summa, означает обращение к подпрограмме с именем Summa, Call - вызов. При структурном подходе к составлению алгоритмов и программ используются три основные типа алгоритмов: условные, циклические алгоритмы и подпрограммы.

*Структурированными* считаются алгоритмы и программы, составленные с использованием только этих трех типов алгоритмов, при этом для записи циклов и условий должна использоваться *ступенчатая запись*. Например:

*Если* a>b *то*

*Вывод* “Первое число больше”

*Иначе*

*Вывод* “Второе число больше”

*Конец если*

Алгоритм считается *неструктурированным,* если нет ступенчатой записи или если при создании программы использован оператор *безусловного перехода* ***GoTo – переход к метке***, т. е. структурное программирование – это программирование без ***GoTo.***

Например, рассмотрим 2 фрагмент программ:

НЦ для I:=I1 до I2

S:=S+A(I)

КЦ;

S=0

I2=5

I1=1

**VV**: Ввод A(I1)

S:=S+A(I1)

I1=I1+1

Если I1≤I2 то перейти к метке VV

Вывод S

Анализируя эти фрагменты, можно сказать, что они подготовлены не по правилам структурного программирования, т. к. в первом фрагменте программы отсутствует *ступенчатая запись*, а во втором есть переход *к метке:*

***Нисходящее программирование***

*Основным принципом технологии структурного программирования являетсяяя нисходящее программирование - это программирование с использованием подпрограмм, которое позволяет вести разработку приложения сверху вниз*. Суть такого программирования состоит в том, что сначала выделяются несколько подпрограмм, решающих глобальные задачи, потом каждый из этих модулей разбивается на небольшое число других подпрограмм и так происходит до тех пор, пока вся задача не окажется реализованной. Достоинство такого подхода в том, что небольшие программы легче отлаживать, программа становится более надежной и подпрограммы можно использовать повторно.

**3. Чтение структурированных программ**

Подпрограммы бывают двух видов: процедуры и функции. Процедуры просто выполняют последовательность операторов, а функции вычисляют значение и передают его в главную программу. Подпрограмма – процедура или подпрограмма – функция- это отдельный блок операторов, начинающийся с заголовка и заканчивающийся признаком конца процедуры или функции. Чтобы подпрограмма имела смысл ей надо получить какие-то значения, которые называются параметрами. Параметры, которые принимаются в подпрограмме, описываются в заголовке и называются *формальными.* Например,   
Процедура Summa(*a,b*) – это заголовок подпрограммы - процедуры, *имя* которой Summa, а в скобках указываются *формальные параметры* *a* и *b.*

Обращение из главной программы к процедуре осуществляется по имени подпрограммы-процедуры с перечнем в скобках параметров, которые ей передаются, например, Call Summa(*x,y*) – означает обратиться к процедуре Summa и передать ей параметры *x* и *y,* которые называются *фактическими параметрами*.

Подпрограмма - функция оформляется таким образом:   
Функция Длина( a, b, c, d), где Длина – имя функции, а в скобках указаны формальные параметры. Подпрограмма–функция возвращает только одно значение, которое обязательно присваивается названию функции в теле подпрограммы–функции. Так как функция возвращает значение, то обращение к ней из основной программы может входить в выражение, как операнд.

*При выполнении процедуры или функции формальные параметры временно заменяются на фактические*.

**Задача 1**. *Составить на структурированном языке алгоритм вычисления периметра треугольника, если известны координаты его вершин (треугольник лежит на плоскости)*.

Обозначим координаты вершин xA, yA, xB, yB, xC, yC и ввод их значений осуществим в главной программе. Пусть AB-расстояние между точками A и B, BC-между B и C, AC-между A и С, а Р – периметр. Периметр вычислим по известной формуле Р=AB+BC+AC, а расстояние же между двумя точками вычислим по формуле: . Вычисление расстояния между двумя точками вынесем в подпрограмму-функцию, назовем ее Длина, а формальные аргументы подпрограммы – функции обозначим через *x1, y1, x2, y2*. Тогда для вычисления AB, BC и AB надо три раза обратиться к подпрограмме-функции, передав ей значения фактических аргументов, сначала координаты, например, точек A и B, затем B и C, A и C.

*Составим подпрограмму-функцию:*

Функция Длина(*x1, y1, x2, y2)*



Длина=d

Конец функции

*Основная программа:*

Ввод координат вершин треугольника xA, yA, xB, yB, xC, yC.

P=Длина(xA,yA,xB,yB)+Длина(xB,yB,xC,yC)+Длина(xA,yA,xC,yC)

Вывод P

**Задача 2.** Дан массив целых чисел {Ai}, где i=1,2,3,…,M, причем М=15. Программа вычисляет произведение сумм некоторых элементов этого массива. Какое алгебраическое выражение описывает работу программы? В программе введены следующие константы: G=1; W=12; T=8; L=15.

Программа 15;

Начать функцию

Функция SUMMA(I1,I2);

S=0;

НЦ для I:=I1 до I2

S:=S+A(I)

КЦ;

SUMMA:=S

Конец функции;

Начать программу

Писать (‘Введите значение массива A’);

НЦ для J:=1 до M

Читать (A(J)

КЦ;

P:= SUMMA(G,W)\*SUMMA(T,L);

Писать (‘Произведение равно:’, P:6);

Конец программы.

Варианты ответов:

1.  2.  3.  4. 

***Правильный ответ с номером 2.***

1. **Стратегии решения задач (**Этапы решения задач на ПК**)**

В процессе подготовки задачи (программной системы (ПС)) на ПК можно выделить такие этапы:

1. ***Постановка задачи***

На этом этапе формулируется цель решения задачи, анализируются требования и подробно описывается содержание задачи, выявляются условия, при которых решается задача, а также определяются входные параметры, которые называются исходными данными. *Например, для задачи 1, рассмотренной в предыдущей теме, целью является вычисление периметра треугольника, а исходными данными являются координаты его вершин, при этом условием является то, что вершины треугольника лежат на плоскости.*

1. ***Формальное построение модели задачи***

На этом этапе составляется формальная модель решения задачи, например, модель базы данных, адекватная оригиналу, модель объектов и потоков информации. *Для задачи 1 мы определили, что периметр треугольника вычислим, если будем знать длины его сторон, а длины сторон определяются по координатам вершин треугольника.*

1. ***Построение математической модели решения задачи***

Этот этап иначе называют *формализацией* задачи, на котором описательная модель записывается с помощью какого-либо формального языка, например, математического. *Для задачи 1, рассмотренной в предыдущей теме, для вычисления периметра используем формулу Р=AB+BC+AC, а для вычисления длины одной стороны такую - .*

1. ***Построение алгоритма***

Процесс обработки данных разбивается на отдельные самостоятельные блоки и определяется последовательность выполнения этих блоков. *Для задачи 1 мы вынесли в самостоятельный блок (в подпрограмму-функцию) вычисление расстояния между двумя точками, вызов которой будет осуществляться из основной программы.*

1. ***Составление программы***

На этом этапе алгоритм записывается на каком - либо конкретном языке программирования.

1. ***Отладка программы***

Когда программа составлена, говорят, что готова ***альфа-версия***. Начинается отладка программы. ***Отладка программы - это процесс поиска и устранения синтаксических и логических ошибок в программе***. Метод выявления ошибок называется ***тестированием***.

Если программная система сложная, и ее разрабатывает *компания-исполнитель* по заданию заказчика, то программа, так называемая ее *альфа-версия,* сначала тестируется людьми компании-исполнителя (тестологами).

По истечении определенного времени тестирования, когда количество ошибок резко уменьшается, начинается интенсивное использование системы у *заказчика* с целью выявления и устранения максимального количества ошибок перед выходом системы на рынок. Этот процесс называется ***бета-тестированием***.

В том случае, если ошибки не найдены, говорят, что при тестировании системы получен ***положительный результат***.

Если заказчик удовлетворен качеством программного продукта, то наступает период его внедрения в эксплуатацию. После того, как заказчик подписывает акт приемки, проект разработки программной системы считается завершенным. Как правило, сотрудничество исполнителя по обслуживанию системы с заказчиком продолжается, которое называется *сопровождением системы*.

***Жизненным циклом*** разработанного программного продукта является весь период от анализа требований до внедрения и сопровождения, т. е. весь период разработки и эксплуатации программного средства.

Главным качеством любого программного продукта является его ***надежность***.

Постановка задачи, формальное построение модели задачи и построение математической модели решения задачи,построение алгоритма – это этапы ***системного*** анализа задачи (если задача сложная, то рассматривают такие *этапы конструирования ИС -разработка архитектуры ИС, разработка структур программ и разработка схемы информационных обменов ИС*), а далее идет этап ***кодирования*** алгоритма.