**2. C Operators**

**2.1 How to write a simple practical program?**

**ఏ**  లాంగ్వేజ్‌లో నయినా input తీసుకొని, వాటితో కాలిక్యులేషన్స్ చేసి, ఫలితాలను స్క్రీన్‌పై display చేసే ప్రోగ్రామింగ్ వ్రాసేటప్పుడు నాలుగు ముఖ్యమైన విషయాలను గుర్తుంచుకోవాలి అవి:  
a) Variable declaration

b) Data reading  
c) Calculations  
d) Printing Results

**1) Variables declare చేయడం**

            ప్రోగ్రాం రన్ అవుతున్నప్పుడు RAMను కంప్యూటరు వాడుకుంటుంది. మనం ప్రోగ్రాం రాసేటప్పుడు ఇచ్చిన సంఖ్యలతోపాటు కాలిక్యులేట్ చేయగా వచ్చిన సంఖ్యలను కూడా నిక్షిప్తం (Store) చేయాల్సి ఉంటుంది. వీటికి మెమరీ కేటాయించడానికి (allocate) variables ను declare చేస్తాం. ఉదాహరణకు, గణితంలో మనం కనుక్కోవాల్సిన దాన్ని x అనుకొని సాధిస్తాం. ఇక్కడ xని variable అంటాం. అలాగే programmingలో కూడా variablesను వాడతాం. ఇలా variablesని declare చేయడం అనేది అన్ని programing లాంగ్వేజీల్లోనూ కీలకమైన అంశం.

|  |  |
| --- | --- |
| సీ లాంగ్వేజీలో తీసుకునే variable namesలో Upper case, Lower case, 0 - 9, Underscore (\_) అక్షరాలను మాత్రమే వాడతాం. ఈ విషయంలో ఒక్కో programming లాంగ్వేజీకి ఒక్కోవిధంగా నియమాలు ఉంటాయి. సీ లాంగ్వేజీలో తీసుకునే variableలో మొదటి అక్షరం Upper Case, Lower Case, Underscoreలలో ఏదో ఒకదాన్ని మాత్రమే ఉపయోగించాలి. |  |

అంతేకాకుండా మనం తీసుకునే variable names, C లాంగ్వేజీలో ఉండే reserved words కాకూడదు. ఉదాహరణకు ప్రోగ్రాంలో తీసుకునే variable పేరు Main, include లుగా ఉండకూడదు. అంతేకాకుండా, ఈ variable ను ఒక్కసారి మాత్రమే declare చేయాలి (ఒకసారి డిక్లేర్ చేసిన variableని ప్రోగ్రాంలో ఎన్నిసార్లయినా వాడుకోవచ్చు). అన్ని variablesనూ, block beginingలో మాత్రమే declare చేయాలి. అంటే, Opening Curly braces ({) తర్వాత మాత్రమే declare చేయాలి.

**ఉదాహరణ:** ప్రోగ్రాంలో వాడగలిగిన Variable Names:

**Salary, Profit, My\_salary, Total\_Income**ప్రోగ్రాంలో వాడకూడని variable Names:   
**Rama.rao** (full stop ఉండకూడదు)  
**my-salary** (minus ఉండకూడదు)  
**int** (reserved variable):

చాలా సీ లాంగ్వేజ్ కంపైలర్లలో, తీసుకొనే variable పేరు పదకొండు (11) అక్షరాల కంటే ఎక్కువగా ఉండకూడదు. కానీ ఇటీవల వస్తున్న సీ కంపైలర్లు పెద్ద variable పేర్లను కూడా support చేస్తున్నాయి. ఒక variableను డిక్లేర్ చేసినప్పుడు అందులో ఎలాంటి సంఖ్యలను నిక్షిప్తం చేయాలనుకుంటున్నామో కూడా తెలపాలి. దీన్నే variableకు typeను ఇవ్వడం అంటాం. ఏదైనా variableలో పూర్ణ సంఖ్యలను store చేయాలంటే int type అని, పెద్ద పూర్ణ సంఖ్యలను store చేయాలంటే long type అని డిక్లేర్ చేయాలి. అలా కాకుండా వాస్తవ సంఖ్యలను (రియల్ నెంబర్స్) స్టోర్ చేయాలంటే float type అని డిక్లేర్ చేయాలి. సాధారణంగా float type అని రాస్తే కాలిక్యులేషన్‌లో 6th decimal వరకూ ఉపయోగించుకోగలుగుతాం. ఇంకా accurateగా కావాలంటే, variableను double type అని డిక్లేర్ చేయాలి. ఈ సందర్భంలో కాలిక్యులేషన్స్ పదకొండు డెసిమల్ స్థానాల వరకూ వస్తాయి. ఒకవేళ variableలో అక్షరాలను store చేయాలంటే, ఆ variableను character type అని declare చేయాలి. ము0దు పేజీలలో వీటన్ని0టి గురు0చి యి0కా డీటైల్ గా తెలుసుకొ0దాము.

ప్రోగ్రాంలో memory allocate చేయాలంటే రెండు విషయాలు గుర్తుంచుకోవాలి.  
a. variable name  
b. variable type

ఉదాహరణ: ముగ్గురు విద్యార్థులు పదోతరగతిలో సాధించిన మార్కులను నిక్షిప్తం చేయాలంటే మూడు variablesను int typeగా డిక్లేర్ చేయాలి. ఎందుకంటే మార్కులు పూర్ణ సంఖ్యలు. ఈ మూడు variablesను కిందివిధంగా డిక్లేర్ చేయాలి.

**int a, b, c;**

పై statementను variable declaration statement అని అంటారు. లైన్ చివరన ';' (సెమీకొలన్)ను ఉంచడాన్ని మరిచిపోవద్దు.

            సీ లాంగ్వేజ్‌లో ప్రతి Executable Statement (మెషిన్ లాంగ్వేజ్‌లోకి translate అయ్యేది)కు చివరన ';' (సెమీకొలన్) ఉండాలి. కంప్యూటరు పైన చెప్పిన statements ని చూసినప్పుడు, మూడు పూర్ణ సంఖ్యలను store (నిక్షిప్తం) చేయడానికి ఎంత మెమరీ కావాలో అంతే కేటాయిస్తుంది.

            అదేవిధంగా ఒక గది పొడవు, వెడల్పు, ఎత్తును నిక్షిప్తం చేయాలంటే మూడు float type variables అవసరం. ఎందుకంటే, ఈ కొలతలు ఎప్పుడూ పూర్ణ సంఖ్యలుగా ఉండకపోవచ్చు. float type variablesను, కిందివిధంగా రాస్తాం.

float L, B, H;  
or  
float Length, Breadth, Height;  
or  
Float My\_Room\_Length, My\_Room\_Breadth, My\_Room\_Height;

ఏ type variableకు ఎంత మెమరీ allocate అవుతుందో ఈ క్రి0ది టేబుల్ చెపుతు0ది.

|  |  |
| --- | --- |
| Integer | [int](javascript:void(0)) |
| Character | char |
| Floating Point | [float](javascript:void(0)) |
| Double precision floating point | double |
| Void | void |

            ఒకసారి మొమరీ కేటాయించిన తర్వాత, ఆ variable memoryలో ఏముంటుందన్న విషయాన్ని ఎవరూ చెప్పలేరు. మొమరీ కేటాయించిన తర్వాత variablesను Un-initialised variables అంటారు. వాటి valuesను garbage values అంటారు. లేదా God Only Knows Values అని కూడా అంటారు.  Un-initialised variablesను కాలిక్యులేషన్‌లలో ఎప్పుడూ వాడకూడదు. ఒకవేళ వాడినా ఫలితం garbage గానే వస్తుంది. ఒక Variableలోకి meanigful value రీడింగ్ ద్వారా రావచ్చు, direct assignment (int a = 10;) ద్వారా రావచ్చు లేదా కాలిక్యులేషన్స్ ద్వారా రావచ్చు. మనం un-intialised variableను కాలిక్యులేషన్స్‌లో వాడితే, java లాంటి modern languages (5th generation languages)లో errors వస్తాయి. కానీ 'సీ'లో ఇలా రావు. అందువల్లే Javaను strongly typed language అని 'సీ'ని weakly typed language అని అంటారు.

**2) డేటా Read చేయడం**

            డేటా రీడ్ చేసిన తర్వాత వాటిని నిక్షిప్తం చేయడానికి ముందే variable ను declare చేయడం ద్వారా వాటికి మెమరీను కేటాయిస్తాం. ఇప్పుడు ఆ మెమరీలోకి కీబోర్డు ద్వారా data values ఎలా రీడ్ చేయాలో తెలుసుకుందాం. దీని కోసం scanf అనే readymade function ను (printf లా) ఉపయోగించాలి. దీన్ని వాడుకోవాలంటే, ముందుగా #include <stdio.h> అనే statement ను program లో రాయాలి. ప్రోగ్రాం రాసేటప్పుడు scanf function కు రెండు విషయాలు యివ్వాలి. అవి:

a. కీ బోర్డు నుంచి ఎన్ని values, ఎలాంటి values రీడ్ చేయాలి. ఈ విషయాన్ని చెప్పడానికి, format string ను ఉపయోగిస్తాం.

**ఉదాహరణ:** కీబోర్డు నుంచి మూడు పూర్ణసంఖ్యలను రీడ్ చేయమని చెప్పడానికి, మనం ''%d%d%d" ను scanf కు మొదటి argument గా ఇస్తాం. అదే నాలుగు పూర్ణ సంఖ్యలు, రెండు వాస్తవ సంఖ్యలను రీడ్ చేయమని చెప్పడానికి "%d%d%d%d%f%f" అని రాస్తాం. ఇక్కడ %d అంటే పూర్ణసంఖ్య అని, %f అంటే వాస్తవ సంఖ్య అని scanf అర్థం చేసుకుంటుంది. అలాగే, %ld, %df, %c లను long, double, character type లకు ఉపయోగిస్తాం.

b. రీడ్ చేసిన విలువలను RAM లో ఎక్కడ పెట్టాలో కూడా scanfకు మనం చెప్పాలి. అంటే RAM address లను ప్రోగ్రాంలో ఇవ్వాలి. C లాంగ్వేజ్‌లో ఒక variableకు ముందు ampersand (&) symbol పెడితే, ఆ variableకు కేటాయించిన RAM memory cell number (or) addressను తెలియజేస్తుంది.

            ప్రోగ్రాంలో కేటాయించిన memory cell numbersను మాత్రమే scanf లో గానీ, ఇంకా ఎక్కడకైనా గానీ వాడాలి. అలా కాకుండా వేరే అంకెలను వాడితే ప్రోగ్రామ్ రన్ అవుతున్నప్పుడు, run-time error (bug) వస్తుంది.  
**ఉదాహరణ:** scanf మూడు పూర్ణ సంఖ్యలను రీడ్ చేసి a, b, c అనే మూడు variables లో పెట్టమని చెప్పడానికి వాటి addressలను ఇవ్వాలి. ఉత్తరంపై ఎవరి చిరునామా రాస్తే వారికే వెళుతుంది. ఇక్కడ కూడా అదే నియమం వర్తిస్తుంది. scanf statement మూడు integersను కీబోర్డు నుంచి రీడ్ చేసి, a, b, c అనే variablesలో పెట్టడానికి కిందివిధంగా రాస్తాం.

**scanf ("%d%d%d" &a, &b, &c);**

అదేవిధంగా amount, rate, time అనే variablesలోకి మూడు float valuesను రీడ్ చేయడానికి  
scanf ("%f%f%f", &amount, &rate , &time); అని రాస్తాం.

**3) Calculations చేయడం**

            రీడ్ చేసిన values తో ఏం కాలిక్యులేట్ చేయాలనేది మన అవసరంపై ఆధారపడి ఉంటుంది. ఏ Operators ఉపయోగించాలి? వాటి అర్థాలేమిటి? తదితర అంశాల గురించి తర్వాతి పేజీలలో తెలుసుకుందాం.

**4) Results Print చేయడం**

            ప్రోగ్రాం పూర్తయిన తర్వాత కాలిక్యులేట్ చేసిన ఫలితాన్ని program screen పై Print చేసుకోవచ్చు. Program కరెక్ట్‌గా ఉందా లేదా అనేది Print చేయడం ద్వారానే తెలుస్తుంది. ఉదాహరణకు రేపు పరీక్ష ఉందనుకోండి. ఈ రోజు రాత్రి మనం బాగా చదివాం. అదృష్టం కొద్దీ చదివిన ప్రశ్నలే వచ్చాయి. జవాబులు తెలిసినా సమాధానపత్రంలో రాయలేదు. అప్పుడు మనకు ఎన్ని మార్కులు వస్తాయి?

**సున్నా.**ఎందువల్ల?  
మనం రాయలేదు కాబట్టి.

            అప్పుడు మీరు నాకు జవాబు తెలుసు కదా.. అయినా మార్కులు ఎందుకు వేయలేదు అనుకుంటే కుదరదు. మీకు సమాధానం తెలుసని మూల్యాంకనం చేసేవారికి మీరు రాస్తేనే కదా తెలిసేది. అదేవిధంగా Program కూడా మనం అడిగితేనే, ఫలితాలను screenపై Print చేస్తుంది. ప్రోగ్రాంకు సంబంధించిన సమాధానాలు RAMలో ఉంటాయి. కానీ మనం అడిగితేనే screen పై Print చేస్తుంది. కాబట్టి తెరపై Print చేయడానికి ఉపయోగించే Printing statements కు ప్రాధాన్యం ఉంది.

            ప్రోగ్రాం ఔట్‌పుట్‌లో variable valuesను Print చేయాలంటే scanf లా format string ఇవ్వాలి. దీని కోసం variables పేర్లను రాస్తే సరిపోతుంది. వాటి address లను ఇవ్వాల్సిన అవసరం లేదు. ఇది print function requirement (దీన్ని అభివృద్ధి చేసినవారు, అలా రూపొందించారు.)

**ఉదాహరణ:**  
\* a, b, c అనే మూడు integer variables విలువలను తెరపై print చేయడానికి కింది printf statementలను ఉపయోగించాలి.  
**printf("%d %d %d\n", a, b, c);**\* విలువలకు మధ్యలో tabకావాలంటే, ఏ '\t' format stringతో కింద చూపినవిధంగా వాడవచ్చు.  
**printf("%d\t %d\t %d\n", a, b, c);**\* ఇలా కాకుండా, మూడు variable values తెరపై ఒక్కోలైన్‌లో రావాలంటే  
**printf("%d\n %d\n %d\n", a, b, c);**\* format stringలో messageలు కూడా రాసుకోవచ్చు.తెరపై "a, b, c values are =" values అని రావాలంటే, printf ను ఇలా రాయాలి.  
**printf("a, b, c values are = %d, %d, %d\n", a, b, c);**\* అదేవిధంగా a, b ల విలువలను print చేయడానికి  
**printf("a = %d, b = %d, c = %d\n", a, b, c); అని రాయాలి.**మొదటి %d దగ్గర a విలువ, రెండో %d దగ్గర b విలువ print అవుతాయి. ఉదాహరణకు a, b, c విలువలు 10, 20, 30 అనుకుంటే, తెరపై కిందివిధంగా ఫలితం వస్తుంది.  
            a = 10, b = 20, c = 30  
ఈవిధంగా printfను వాడుకొని screenపై ఫలితాలను print చేసుకోవచ్చు.

ఇప్పటి వరకూ మనం నేర్చుకున్న అంశాల ఆధారంగా ఒక పూర్తి ప్రోగ్రాం రాసే విధానాన్ని పరిశీలిద్దాం.  
a) ఎన్ని values read చేయాలి? ఏం కాలిక్యులేట్ చేయాలి? అనే అంశాల ఆధారంగా ముందుగా variablesను డిక్లేర్ చేయాలి.  
b) scanfను ఉపయోగించి డేటా read చేయడం.  
c) ఇచ్చిన ప్రోగ్రాం కోడ్ ఆధారంగా రీడ్ చేసి వాటితో కాలిక్యులేషన్స్ చేయాలి.  
d) కాలిక్యులేట్ చేసినవాటిని screenపై రాయమని printf ద్వారా చెప్పాలి.

     పైన చెప్పిన స్టేట్‌మెంట్లతోపాటు ప్రోగ్రామర్‌కు కొన్ని సూచనలు చేయడానికి instructionsను వాడతాం. వీటిని user interface statements అంటారు. ఇవి లేకపోయినా program పని చేస్తుంది. కానీ ఒకరు రాసిన ప్రోగ్రాం మరొకరు పరిశీలించినా అర్థమవడానికి ఇవి ఉపయోగపడతాయి.

**ఉదాహరణ:** మూడు పూర్ణసంఖ్యలను (Integer Values)కీబోర్డు నుంచి enter చేయమని చెప్పడానికి scanf statement ముందు కింది స్టేట్‌మెంట్‌ని రాస్తాం.

**printf("Enter three integer values\n");**

ఇలాంటివాటినే user interface statements అంటారు.

**Example 1:**            ఇప్పుడు మనం ఒక చిన్న ప్రోగ్రాం రాద్దాం. ముగ్గురు విద్యార్థుల మార్కులను కీబోర్డు నుంచి read చేసి వాళ్ల మార్కుల సరాసరిని తెరపై print చేయాలనుకుందాం.  
i) దీని కోసం మూడు int type variables తోపాటు ముగ్గురి మార్కులను నిక్షిప్తం చేయడానికి ఒక float variable కావాలి. ఎందుకంటే సరాసరి ఎప్పుడూ పూర్ణ సంఖ్యగా ఉంటుందని చెప్పలేం. మొత్తం మీద నాలుగుvariables అవసరం.

ii) Enter three students marks అని User Interface statement రాయాలి.

iii) scanfకు ముందు integersను కీబోర్డు నుంచి రీడ్ చేసి variablesలో నిక్షిప్తం చేయమని చెప్పాలి. ఒకవేళ నాలుగు variables కు value చేస్తే ఎలా ఉంటుంది? ప్రశ్నతోపాటు సమాధానం ఇచ్చినట్లు ఉంటుంది. సరాసరిని మనం కాలిక్యులేట్ చేయాలి. ఏ మూడు మార్కులు ఇచ్చినా, మనకు వాటి సరాసరి కావాలి.

iv) మనం ఇచ్చిన మూడు విలువల సరాసరిని కాలిక్యులేట్ చేయమని చెప్పే statement రాయాలి.

v) calculate చేసిన సరాసరిని screenపై ప్రింట్ చేయమని చెప్పాలి.

ఈ సూచనలను పాటిస్తూ రాసిన ప్రోగాం, దాని output కింద చూపిన విధంగా ఉంటుంది..

**#include<stdio.h>**

**int main(){**

**int a,b,c; /\* Variable declarations. \*/**

**float avg;**

**/\* User interface statment\*/**

**printf("Enter three students marks\n");**

**/\* Data reading statement \*/**

**scanf("%d%d%d",&a,&b,&c);**

**/\* Calculation of average \*/**

**avg = (a+b+c)/3;**

**/\* Statements to print results\*/**

**printf("%f \n", avg);**

**printf("Average of %d %d %d = %f\n",a,b,c,avg);**

**printf("%f is the average of %d, %d %d\n", avg,a,b,c);**

**//**

**return 0;**

**}**

**Output:**

**Enter three students marks**

**10 30 90**

**43.000000**

**Average of 10 30 90=43.000000**

**43.000000 is the average of 10, 30, 90**

          ప్రోగ్రాము రన్ అవుతున్నప్పుడు విలువల మధ్యలో space, TAB, new line లలో ఏదో ఒకటి ఇవ్వాలి. ఇక్కడ ఇచ్చిన Programలో మూడు printf statementలను చివరలో ఉపయోగించాం. మనకు screenపై ఏవిధమైన ఫలితం కావాలో, ఆవిధంగా printfను ఉపయోగించవచ్చు. అంతేగానీ మూడూ ఒకేసారి ఉపయోగించాల్సిన అవసరం లేదు.

**2.2 Operators**

**2.2.1. Integer mode computations**

**Example 2**: ‚¡ÁôåžÁÅ Farenheit temparatureþÁÅ ‚þ÷ ¡Áôý÷ ¨ÂÂ œÄ¬ÁÅÌþÃ centigrade temparature ¨ÍþÃÃ ¥Á Â§ÊÖ program ©ÂëžÂâÏ.

Solution: ‡þÃä variables Â©Â¨Ã?. §ÉÏ™ÁÅ Â©Â¨Ã. ŠÁýÃ read úÊ¬ÃþÁžÂþÃþÃ store úÉ¦Áê™ÂþÃÃ, §ÉÏ™ÍžÃ ¥ÁþÁ¥ÁÅ calculate úÊ¬ÃþÁ žÂþÃþÃ store úÉ¦Áê™ÂþÃÃ. €ÏžÁÅÁþÃ ƒ ÃëÏžÁ program ¨Í §ÉÏ™ÁÅ variables it, ot ¨þÁÅ ©Â™Â¥ÁÅ. ³ÂŸÂ§Á›ÏÂ temparature values real Â ©ÁôÏýÂ¦ ÁþÁÅÁ ƒ variablesþÁÅ float typeÂ declare úÊ¬ÃþÂ¥ÁÅ. ™ÂýÂ read úÉ¦Áê™ÂþÃÃ scanf ¨Í %f ©Â™Â¥ÁÅ.

**#include<stdio.h>**

**int main(){**

**float it, ot;**

**printf(“Enter Fahrenheit temperature value”);**

**scanf(“%f”, &it);**

**ot = 5/9\*(it-32);**

**printf(“%f\n”, ot);**

**return(0);**

**}**

ƒ program compile, run ÁÆ™Â €©ÁôœÁÅÏžÃ, ÂþÃ ˆ ™ÂýÂ ©Â¨Æê ‚úÃÖþÂ ¬ÁÅþÂä ©Á¬ÁÅàÏžÃ. ‡ÏžÁÅ©Á¨þÁ €þÁÂ computers¨Í calculations úÊ¬ÁÅàþÁä¡Áôå™ÁÅ ‹ operator ¦ÉÅÁÑ §ÉÏ™ÁÆ operands integers €¦œÊ result ÁÆ™Â integer¨Í ©Á¬ÁÅàÏžÃ. €þÁÂ 5/9 value ¬ÁÅþÂä €©ÁôœÁÅÏžÃ. €ÏžÁÅ ©Á¨þÁ, ¥ÉÅœÁàÏ expression value ¬ÁÅþÂä €©ÁôÏœÁÅÏžÃ. ‚¨Â ÂÁÅÏ™Â ©ÁôÏ™Á™ÂþÃÃ ot=5/9\*(it-32); €þÊ statementþÁÅ ƒ ÃëÏžÃ ©ÂýÃ¨¨Í žÊþÃœÍþÁ¦þÂ replace úÉ¦Áê©ÁúÁÅÖ. ‡ÁÑ™Á (float) 5 or (float) 9 €þÊ ©ÂýÃþÃ typecasting €ÏýÂ§ÁÅ. ఇ©Ã 5 þÁÅ 5.0 Â, 9 þÁÅ 9.0 Â œÄ¬ÁÅÌÏýÂ¦.

**ot=5\*(it-32)/9; OR ot = 5.0/9\*(it-32); OR ot = 5/9.\*(it-32);**

**ot = 5.0/9.\*(it-32); OR ot = (float)5/9\*(it-32); OR**

**ot = 5/(float)9\*(it-32); OR ot =(float) 5/(float)9\*(it-32);**

ఇ0తకు ము0దు exampleలో integer mode మరియూ float mode computations గురు0చి తెలుసుకొన్నాము. ఓ integer variableకు float variableను assign చేస్తే float variable లోని integer భాగము మాత్రమే integer variableకు assign అవుతు0ది. అలాగే float results రావాల0టే expression యొక్క RHS భాగము calculations float modeలో జరగాలి, మరియు LHS లో float లేక double టఈప్ variable అయి వు0డాలి.

**Example 3**: ఈ ప్రోగ్రాము angleను degrees, minutes, seconds లో ఇస్తే దానికి సమానమయిన angleను radiansలో ప్రి0ట్ చేయాలి. గమనిక: 1800=3.14 radians, 10=60', 1'=60''.

**Solution:** ము0దు టోటల్ angle degrees లో కనుక్కొని 3..14/180తో గుణి0చాలి.

**#include<stdio.h>**

**int main(){**

**int d, m, s;**

**float rad;**

**printf(“Enter angle in deg, min, sec”);**

**scanf(“%d%d%d”, &d, &m, &s);**

**rad = (d+(m/60.0)+(s/3600.0))\*(3.14/180);**

**printf(“%f \n”, rad);**

**return(0);**

**}**

పై ప్రోగ్రాములో 60.0 బదులు 60, 3600.0 బదులు 3600 పెడితే వచ్చే వాల్యూను గమని0చ0డి.

M

**2.2.2 Modulus operator (%)**

ఇది binary operator. అనగా రె0డు integer operands దీనితో వాడుతారు. A,B లు రె0డు operands అయితే, A%B=|A|/|B| యొక్క శేషము(reminder) మొదటి operand యొక్క signతో. అనగా |A|ను |B| divide చెయ్యగా మిగిలినది. ఈ operator యొక్క రిజల్ట్ sign మొదటి operand లాగా వు0టు0ది. ఉదాహరణకు, 10%3=1, -10%3=-1, 10%-3=1, -10%-3=-1. అలాగే, A%B యొక్క వాల్యూ 0 ను0చి B-1 లోపల వు0టు0ది. మరియూ, A కనుక B కన్నా చిన్నది అయితే, A%B యొక్క వాల్యూ A.

**Example 4:** రె0డు పనులు చేయడానికి పట్టే timeను hours, minutes లో తీసుకొని మొత్త0 time ప్రి0ట్ చేయడానికి ప్రోగ్రాము ఇది. దీనికు మనము మొత్తము నాలుగు వాల్యూస్ రీడ్ చేసి టోటల్ hours, minutes ప్రి0ట్ చేయాలి. ము0దుగా రె0డు పనులలోని minutes యొక్క టొటల్ కనుక్కొని, అ0దులో ఎన్ని hours వున్నాయో కనుక్కొని మిగతా రె0డు గ0టలతో కలిపితే మనకు టోటల్ గ0టలు వస్తాయి. అలాగే, టొటల్ minutes ను 60తో modulus అప్లై చేస్తే టోటల్ minutes లోను0చి hours తీసివేసిన తర్వాత మిగిలిన minutes వస్తాయి. ఈ క్రి0ద ఇచ్చిన ఉదాహరణలు, ప్రోగ్రాము చూస్తే మనకు అర్ధము అవుతు0ది.

Ex: Hr Min Hr Min

4 57 4 17

3 32 3 32

7 89 = T 7 49 = T

T/60 = 1; T%60 = 49 T/60 = 1; T%60 = 49

8 29 7 49

**#include<stdio.h>**

**int main()**

T/60 gives number of hours in the total minutes.

**{**

**int h1, m1, h2, m2, T, H, M;**

**scanf(“%d%d%d%d”, &h1, &m1, &h2, &m2);**

**T = m1+m2;**

T%60 gives left over minutes in total minutes after taking out hours from the total minutes.

**H = h1+h2+ T/60;**

**M = T%60;**

**printf(“%d%d\n”, H,M);**

**return(0);**

**}**

**2.2.3 Relational Operators**

ఇవి కూడా binary operators. అనగా వీటికీ రె0డు operands వు0టాయి. ఈ operands ఏ టైప్ అయిన variable అయినా, constant అయినా, expression అయినా కావాచ్చు. వీటి రిజల్ట్స్ ఎప్పుడూ True లేక False లాగా వు0టాయి, దానినే స0ఖ్యాపరముగా 1 లేక 0 అని చెప్పవచ్చు. ఈ క్రి0ది టేబులు వీటిని విశదీకరిస్తు0ది. ఇక్కడ A యొక వాల్యూ 70, మరియూ B యొక్క వాల్యూ 34 అని అనుకొన్నాము.

|  |  |  |  |
| --- | --- | --- | --- |
| Operator | Usage | Logical Value | Numerical value |
| < | A<B | False | 0 |
| <= | A<=B | False | 0 |
| > | A>B | True | 1 |
| >= | A>=B | True | 1 |
| == | A==B | False | 0 |
| != | A!=B | True | 1 |

**Example** 5: ƒ program ‰žÁÅÁÅ§Ã students marks read úÊ¬Ã, ±Â¬÷ €¦þÁ ©Â§Ã average marksను ప్రి0ట్ úÊ¬ÁÅàÏžÃ.

**Solution:**¥ÁÅÏžÁÅÂ, ‰žÁÅÁÅ§Ã మార్క్స్ స్టొర్ úÉ¦Áê™ÂþÃÃ ‰žÁÅ integer variables Â©Â¨Ã. ±Â¬÷ €¦þÁ ©Â§Ã average Â©Â¨ÏýÊ ‡ÏœÁ¥ÁÏžÃ ±Â¬÷ €¦Á Âê§ÁÅ, ©Â§Ã total marks ‡þÍä œÉ¨Ã¦Á Â¨Ã. ‚©Ã ÁÆ™Â integers ¨ÂÁ ©ÁôÏýÂ¦ ÁþÁÅÁ, ¦ÏÍ §ÉÏ™ÁÅ integer type variables €©Á¬Á§ÁÏ €©ÁôœÂ¦. €ÏœÊ ÂÁÅÏ™Â, averageþÁÅ ³ÍÛ§÷ úÉ¦Áê™ÂþÃÃ ‚ÏÍ variable Â¬Â¨Ã. ¥ÁþÁÁÅ average ÁÆ™Â integer ¨ÂÁ ©Á¬Êà úÂ¨Å అని అనుకొనిచున్నాము. ÁþÁÅÁ €þÃä variablesþÁÅ integer type ¨ÂÁ declare úÉ¦Áê©Á¨ÉþÁÅ. ±Â¬÷ €©Á™ÂþÃÃ minimum 35 మార్క్స్ §Â©Á¨ÉþÁÅ. ‡ÏœÁ¥ÁÏžÃ ±Â¬÷ €¦Á Âê§Í œÉ¨Å¬ÁÅÍ©Á™ÂþÃÃ, ¡ÁëœÃ student మార్క్స్þÃ 35œÍ relational operator >= ©Â™Ã compare úÊžÂâ¥ÁÅ. ±Â¬÷ €¦œÊ 1, fail €¦œÊ 0 ‚¬ÁÅàÏžÃ ‚žÃ. ©ÄýÃ total ¥ÁþÁÁÅ ‡ÏœÁ¥ÁÏžÃ ±Â¬÷ €¦Áê§Í ‚¬ÁÅàÏžÃ. €¨ÂÊ, ‹ relational operator result þÁÅ ‹ variableœÍ ÁÅ›Ã¬Êà,  relational operator result true €¦œÊ  variable value final result €©ÁôœÁÅÏžÃ, ¨ÊÁ±ÍœÊ ¬ÁÅþÂä €©ÁôœÁÅÏžÃ. ƒ logic þÁÅ ©Â™ÁÅÁÅÏýÆ, ±Â¬÷ €¦þÁ ©Â§Ã average marks ÁþÁÅÍÑ©Á™ÂþÃÃ, ±Â¬÷ €¦þÁ ©Â§Ã total మార్క్స్ þÁÅ ‡ÏœÁ¥ÁÏžÃ ±Â¬÷ €¦Á Âê§ÌœÍ divide úÊžÂâ¥ÁÅ.  œÁ§ÂíœÁ యావరేజిþÁÅ print úÊžÂâ¥ÁÅ. ƒ ÃëÏžÁ ¡Áõ§Ãà program ‚©Áí£™ÃþÁžÃ.

**#include<stdio.h>**

**int main(){**

**int a, b, c, d, e, np, sp, avg;**

**printf(“Enter 5 students marks\n”);**

**scanf(“%d%d%d%d%d”, &a, &b, &c,&d, &e);**

**np = (a> = 35)+(b> = 35)+(c> = 35)+(d> = 35)+(e> = 35);**

**sp =(a>=35)\*a+(b>=35)\*b+(c>=35)\*c+(d> = 35)\*d+(e>= 35)\*e;**

**avg = sp/np;**

**printf(“%d\n”,avg);**

**return(0);**

**}**

ప్రోగ్రాము §Áþ÷ €©ÁôœÁÅþÁä¡ÁôåžÁÅ, ¥ÁþÁ¥ÁÅ 70 16 90 31 40 ‚¬Êà, ¥ÁþÁÁÅ యావరేజి 66 ¨ÂÁ ©Á¬ÁÅàÏžÃ. ‡¨ÂÁ €þÁÂ, np = (a> = 35)+(b> = 35)+(c> = 35)+(d> = 35)+(e> = 35); వాల్యూ 1+0+1+0+1 €©ÁôœÁÅÏžÃ. ¥Á§Ã¦ÁÆ, sp =(a>=35)\*a+(b>=35)\*b+(c>=35)\*c+(d> = 35)\*d+(e>= 35)\*e; వాల్యూ 1\*70+0\*16+1\*90+1\*31+1\*40=200. €ÏžÁÅ©Á¨þÁ, యావరేజి 200/3, 66 €©ÁôœÁÅÏžÃ. €దేÊ ¥ÁþÁ¥ÁÅ 31 16 31 27 19 ‚¬Êà, Divided by zero €þÊ run time error ©Á¬ÁÅàÏžÃ, ‡ÏžÁÅÁþÁÂ, ‡ÏœÁ¥ÁÏžÃ ±Â¬÷ €¦Á Âê§ÁÅ, ©Â§Ã total ÁÅ™Â 0 €©ÁôœÂ¦, ÁþÁÅÁ average 0/0 €©ÁôœÁÅÏžÃ. €ÏžÁÅ©Á¨þÁ error ©Á¬ÁÅàÏžÃ. 0/0 €þÊžÃ mathematics¨Í ‡ÁÑ™Â define úÉ¦Áê¨ÊžÁÅ. ƒ errorþÁÅ ¥Ä§ÁÅ œÁ¡ÁåÁ úÁÆ™Â¨Ã.

**2.2.4 Conditional operator or Compact if operator**

žÄþÃþÃ ƒ ÃëÏžÁ ‚úÃÖþÁ ©ÃŸÂ¨ÅÂ ©Â™Á©ÁúÁÅÖ.

**1. (expr)?expr1:expr2;**

**2. var=(expr)?expr1:expr2;**

‚ÁÑ™Á, expr, expr1, expr2¨Å variables €¦þÂ, constants €¦þÂ, expressions €¦þÂ Â©ÁúÁÅÖ. ¥Á§Ã¦ÁÆ var €ÏýÊ ˆ variable €¦þÂ €þÃ ‚ÁÑ™Á €§Áã¥ÁÅ. ¥ÉÅžÁýÃ స్టైల్¨Í, expr value true €¦œÊ expr1þÁÅ calculate úÊ¬ÁÅàÏžÃ, ¨ÊÁ±ÍœÊ expr2þÁÅ calculate úÊ¬ÁÅàÏžÃ. €¨ÂÊ, §ÉÏ™Í స్టైల్ ¨Í, expr value true €¦œÊ expr1þÁÅ calculate úÊ¬Ã ©ÁúÃÖþÁ valueþÁÅ varÁÅ ‚¬ÁÅàÏžÃ, ¨ÊÁ±ÍœÊ expr2þÁÅ calculate úÊ¬Ã ©ÁúÃÖþÁ valueþÁÅ varÁÅ ‚¬ÁÅàÏžÃ.

**Example 6**: ‹ integerþÁÅ read úÊ¬Ã €žÃ even þÁÏ£§Â ¨ÊÁ odd þÁÏ£§Â €þÊžÃ print úÉ¦Á Âê¨Ã.

**#include<stdio.h>**

**int main(){**

**int n;**

**printf("Enter an integer\n");**

**scanf("%d",&n);**

**(n%2)?printf("Even\n"):printf("Odd\n");**

**return (0);**

**}**

ƒ program §Áþ÷ úÊ¬ÁÅàþÁä¡Áôå™ÁÅ, ¥ÁþÁ¥ÁÅ 71 input ¨ÂÁ ‚¬Êà, 71%2 value 1 €©ÁôœÁÅÏžÃ. C language¨Í positive valueþÁÅ true ¨ÂÂ œÄ¬ÁÅÁÅÏýÅÏžÃ. €ÏžÁÅ©Á¨þÁ, printf("Even\n") €þÊ statementþÁÅ execute úÊ¬ÁÅàÏžÃ, ÁþÁÅÁ ¥ÁþÁÁÅ Even €þÊžÃ ¬ÄÑëþ÷ ¥ÄžÁ ©Á¬ÁÅàÏžÃ. €žÊ ¥ÁþÁ¥ÁÅ 80 input ¨ÂÁ ‚¬Êà, 80%2 value 0 ÁþÁÅÁ, system žÂþÃþÃ false €þÁÅÌþÃ, printf("Odd\n") €þÊ statementþÁÅ execute úÊ¬ÁÅàÏžÃ. €ÏžÁÅ©Á¨þÁ ¥ÁþÁÁÅ ¬ÄÑëþ÷ ¥ÄžÁ, Odd €þÃ ©Á¬ÁÅàÏžÃ.

**Example 7**: ‹ student ¦ÉÅÁÑ ‹ test marks input ¨ÂÁ œÄ¬ÁÅÌþÃ, ±Â³Â, ¢É¦¨Â €þÃ ‚©Áí™ÁþÃÃ, program §ÂžÂâ¥ÁÅ.

**Solution**: ‹ integer variableþÁÅ declare úÊ¬Ã, žÂÏýÍìÃ ‹ value read úÊ¬Ã žÂþÃþÃ 35 œÍ compare úÊ¬Ã, €žÃ true €¦œÊ Passed €þÃ print úÊ¬ÊýýÅì, ¨ÊÁ±ÍœÊ Failed €þÃ print úÊ¬ÊýýÅì ƒ ÃëÏžÁ program conditional operatorþÁÅ ©Â™ÁÅÌþÃ ©ÂëªÂ¥ÁÅ.

**#include<stdio.h>**

**int main(){**

**int n;**

**printf("Enter a student marks in a test\n");**

**scanf("%d",&n);**

**(n>=35)?printf("Passed\n"):printf("Failed\n");**

**return (0);**

**}**

ƒ program §Áþ÷ úÊ¬ÁÅàþÁä¡Áôå™ÁÅ, ¥ÁþÁ¥ÁÅ 71 input ¨ÂÁ ‚¬Êà, 71>=35 value true €©ÁôœÁÅÏžÃ, ÁþÁÅÁ ¥ÁþÁÁÅ Passed €þÊ ¥É¬ÉðüÃ ¬ÃÑëþ÷ ¥ÄžÁ ©Á¬ÁÅàÏžÃ. €žÊ ¥ÁþÁ¥ÁÅ, program §Áþ÷ úÊ¬ÁÅàþÁä¡Áôå™ÁÅ, ¥ÁþÁ¥ÁÅ 17 input ¨ÂÁ ‚¬Êà, 17>=35 value false €©ÁôœÁÅÏžÃ, ÁþÁÅÁ ¥ÁþÁÁÅ Failed €þÊ ¥É¬ÉðüÃ ¬ÃÑëþ÷ ¥ÄžÁ ©Á¬ÁÅàÏžÃ,

Assignment: ‹ student ¦ÉÅÁÑ 5 test marks input ¨ÂÁ œÄ¬ÁÅÌþÃ, ±Â³Â, ¢É¦¨Â €þÃ ‚©Áí™ÂþÃÃ program §Â¦ÁÏ™Ã.

Hint: ±Â¬÷ €©Âí¨ÏýÊ, ¡ÁëœÄ test¨Í ±Â¬÷ €©Âí¨Ã, ˆ test¨Í þÁ¦þÂ ±Â¬÷ Â©Á™ÂþÃÃ minimum 35 §Â©Â¨Ã. ÁþÁÅÁ, ¥Ä§ÁÅ ‚úÃÖþÁ 5 test marks¨Í ‡þÃä 35 ÁþÂä ‡ÁÅÑ©Í ÁþÁÅÍÑþÃ, ‚žÃ ÁþÁÅÁ 5 €¦œÊ Passed €þÃ ¨ÊÁ±ÍœÊ Failed €þÃ conditional operatorþÁÅ ©Â™ÁÅÌþÃ úÉ¦Áê©ÁúÁÅÖ.

Example 8: ‹ student ¦ÉÅÁÑ 5 test marks input ¨ÂÁ œÄ¬ÁÅÌþÃ distinction ‚©Áí™ÂþÃÃ program §Â¦ÁÏ™Ã. Distinction ‚©Âí¨ÏýÊ, ¡ÁëœÄ test¨Í minimum 60 §Â©Â¨Ã, ¥Á§Ã¦ÁÅ average 75 ©ÁôÏ™Â¨Ã.

**Solution**: žÄþÃþÃ solve úÉ¦Áê™ÂþÃÃ, ‡þÃä test¨¨Í 60 ÁþÁä ‡ÁÅÑ©Á ©ÁúÂÖ¦ÉÂ relational operator >= œÍ ÁþÁÅÌÑþÃ, np €þÊ variable¨Í store úÊžÂâ¥ÁÅ. €¨ÂÊ, total marks ÁþÁÅÌÑþÃ, ©ÂýÃþÃ s ¨Í ³ÍÛ§÷ úÊžÂâ¥ÁÅ. Average 75 €þÂä, total 375 €þÂä ŠÁýÊ ÁžÂ?. œÁ§ÂíœÁ, np value 5 €¦ÏžÂ ¥Á§Ã¦ÁÆ s value 375 ÁþÂä ‡ÁÅÑ©Â ÂžÂ €þÃ §ÉÏ™ÁÆ úÉ÷ úÊ³Âà¥ÁÅ. §ÉÏ™ÁÆ true €¦œÊ, distinctionÁÅ eligible €þÃ, ¨ÊÁ±ÍœÊ, ¨ÊžÁÅ €þÃ ¬ÄÑëþÁÅ ¥ÄžÁ print úÊ³Âà¥ÁÅ.

**#include<stdio.h>**

**int main(){**

**int a, b, c, d, e, np, s;**

**printf(“Enter 5 test marks\n”);**

**scanf(“%d%d%d%d%d”, &a, &b, &c, &d, &e);**

**np = (a>=60)+(b>=60)+(c>=60)+(d>=60)+(e>=60);**

**s = a+b+c+d+e;**

**((np == 5)\*(s>=375))?pritnf(“Eligible”):printf(“Not Eligible”);**

**return(0);**

**}**

ÃëÏžÁ ‚©Áí£™ÃþÁ table, ƒ program ¦ÉÅÁÑ various possible inputs, ¥Á§Ã¦ÁÆ expect úÊసేÊ resultsþÁÅ illustrate úÊ¬ÁÅàÏžÃ.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| marks | np | s | Eligible? | Reason |
| 90,90,80,90,90 | 1 | 440 | Yes | రె0డు conditions satisfy అవుతున్నాయి. |
| 90,40,90,90,100 | 0 | 410 | No | Min 60 conditions మాత్రమే satisfy అవడములేదు. |
| 60,60,60,60,60 | 1 | 300 | No | Aggregate condition satisfy అవడములేదు. |
| 20,40,60,90,60 | 0 | 270 | No | రె0డు conditions satisfy అవడములేదు. |

**2.2.5 Logical Operators, Logical AND (&&), Logical OR(||)**

‚©Ã ÁÅ™Â binary operators. €þÁÂ, ©ÄýÃ ¦ÉÅÁÑ §ÉÏ™ÁÅ operands ÁÆ™Â variables €¦þÂ, constants €¦þÂ, expressions €¦þÂ Â©ÁúÁÅÖ. Logican AND result, §ÉÏ™ÁÅ operands true €¦œÊþÊ true €©ÁôœÁÅÏžÃ, ¨ÊÁ±ÍœÊ false €©ÁôœÁÅÏžÃ. €¨ÂÊ, Logical OR ¦ÉÅÁÑ result, ˆ ŠÁÑ operand true €¦þÂ true €©ÁôœÁÅÏžÃ, ¨ÊÂ±ÍœÊ false €©ÁôœÁÅÏžÃ.

**Example 9**: Year þÁÏ£§ÁÅþÃ input ¨ÂÁ œÄ¬ÁÅÌþÃ, €žÃ ¨Ä¡÷ year €©ÁôþÍ ÂžÍ print úÉ¦Áê™ÂþÃÃ program §ÂžÂâÏ. ‹ year þÁÅ ¨Ä¡÷ year €þÂ¨Ã €ÏýÊ, ƒ ÃëÏžÃ ©ÂýÃ¨Í ˆžÍ ŠÁ condition satisfy €©Âí¨Ã.

1. year þÁÏ£§ÁÅ 400 úÊœÁ divide €©Âí¨Ã.

2. year þÁÏ£§ÁÅ 4 úÊœÁ divide €©Âí¨Ã ¥Á§Ã¦ÁÆ 100 úÊœÁ divide ÂÁÆ™ÁžÁÅ.

**Solution**:¡Áõ§Ãà program ÃëÏžÁ ‚©Áí£™ÃþÁžÃ. ŠÁ variable ¨ÍÃ year value input ¨ÂÁ œÄ¬ÁÅÌþÃ, Logical OR, Logical AND operator¨þÁÅ ©Â™Ã, program ©ÂëªÂ¥ÁÅ. „žÂÿÁ§Á›ÁÅ, 2000 ÂþÃ 1988 input ¨ÂÁ ‚¬Êà program ¨Ä¡÷ year €þÃ ‚¬ÁÅàÏžÃ. €žÊ 1986 ‚¬Êà, ¨Ä¡÷ year ÂžÁÅ €þÃ ‚¬ÁÅàÏžÃ.

**#include<stdio.h>**

**int main(){**

**int y;**

**printf(“Enter a year number\n”);**

**scanf(“%d”, &y);**

**(y%400==0)||((y%4==0)&&(y%100!=0))?printf(“Leap year\n”):printf(“Not a leap year\n”);**

**return(0);**

**}**

2000 input ¨ÂÁ ‚¬Êà, (y%400==0) €þÉžÃ true €¦þÁÏžÁÅ ©Á¨þÁ Leap Year €þÃ ©Á¬ÁÅàÏžÃ. €žÊ 1988 input ¨ÂÁ ‡¬Êà, ((y%4==0)&&(y%100!=0) €þÉžÃ true €¦þÁÏžÁÅ ©Á¨þÁ Leap Year €þÃ ©Á¬ÁÅàÏžÃ. 1986 ÁþÎÁ input ¨ÂÁ ‚¬Êà, §ÉÏ™ÁÅ conditions ÁÆ™Â false €¦þÁÏžÁÅ ©Á¨þÁ, Not Leap Year €þÃ ©Á¬ÁÅàÏžÃ.

**2.2.6 Unary Increment/Decrement Operators**

ƒ operators mathematics¨Í ¥ÁþÁ¥ÁÅ úÁžÁ©Á¨ÊžÁÅ. ©ÄýÃþÃ integer type variablesÁÅ ¥Á ÂœÁë¥Ê ©Â™Á©ÁúÁÅÖ. ‚ÏÁ ©ÊýÃÄ ©Â™ÁÁÆ™ÁžÁÅ. Postfix increment/decrement operatorþÁÅ ‹ ƒ variableÁÅ ‹ expression¨Í ©Â™ÃþÁ¡Áôå™ÁÅ, žÂþÃ ¡Áë¬ÁÅàœÁ valueœÍ expression evaluate €©ÁôœÁÅÏžÃ,  œÁ§ÂíœÁ, variable value ŠÁýÃ ¡É§ÁÁ™ÁÏ €¦þÂ, œÁÁÓ™ÁÏ €¦þÂ ü§ÁÅÁÅœÁÅÏžÃ. €¨ÂÊ, Prefix increment/decrement operatorþÁÅ ‹ variableÁÅ ‹ expression¨Í ©Â™ÃþÁ¡Áôå™ÁÅ, žÂþÃ value ¥ÁÅÏžÁÅ ŠÁýÃ ¡É§ÁÁ™ÁÏ €¦þÂ, œÁÁÓ™ÁÏ €¦þÂ ü§ÁÅÁÅœÁÅÏžÃ,  œÁ§ÂíœÁ  ¥Á Â§ÃþÁ valueœÍ expression evaluate €©ÁôœÁÅÏžÃ. ƒ ÃëÏžÁ i €þÊ variable ¥ÄžÁ ƒ operatorsþÁÅ ©Â™ÁýÏ úÁÆ™Á©ÁúÁÅÖ.

Operators Postfix Prefix

Increment i++ ++i

Decrement i-- --i

**Example 10**: ƒ ÃëÏžÃ program unary increment, decrement operators ‡¨Â ©Â™Á¨Í úÁÆ¡Ã¬ÁÅàÏžÃ.

**#include<stdio.h>**

**int main(){**

**int i=10,j, k, l;**

**j=i++ + i++; printf(“%d%d\n”, i, j);**

**k = ++ i + ++ i; printf(“%d%d\n”, i, k);**

**l = ++ i + i --; printf(“%d%d\n”, i, l);**

**return(0);**

**}**

Results: i = 12, j = 20; i = 14 , k = 28; i = 14 , l = 30;

j=i++ + i++; ¨Í i+i calculate úÊ¬Ã jÁÅ 20 ‚úÃÖþÁ œÁ§ÂíœÁ i value §ÉÏ™ÁÅ ³Â§ÁÅì ¡É§ÁÅÁÅœÁÅÏžÃ,§ÉÏ™ÁÅ postfix increment¨Å iÁÅ©ÁôþÂä¦ ÁþÁÅÁ. ÁþÁÅÁ €žÃ 22 €©ÁôœÁÅÏžÃ. €žÊ k = ++ i + ++ i; ¨Í, iÁÅ §ÉÏ™ÁÅ prefix increment¨Å ©ÁôþÂä¦ ÁþÁÅÁ i value ¥ÁÅÏžÁÅÂ §ÉÏ™ÁÅ ³Â§ÁÅì ¡É§ÃÃ 14 €¦þÁ œÁ§ÁíœÁ i+i caclulate úÊ¬Ã kÁÅ ‚¬ÁÅàÏžÃ. ÁþÁÅÁ k value 28 €©ÁôœÁÅÏžÃ. €¨ÂÊ, l = ++ i + i --; ¨Í i value ¥ÁÅÏžÁÅÂ ŠÁýÃ ¡É§ÁÅÁÅœÁÅÏžÃ, œÁ§ÂíœÁ i+i value, 15+15 lÁÅ ‚¬ÁÅàÏžÃ, l value 30 €©ÁôœÁÅÏžÃ.  œÁ§ÂíœÁ, i value ¥Á§Á¨ œÁÃÓ 14ÁÅ ©Á¬ÁÅàÏžÃ.

**2.2.7 Mathematical Library Functions**

C ¨ÂÏÊíüÃ úÂ¨Â ready made mathematics ¢ÁÏÁïþÁÅ¨þÁÅ (programs) Á¨ÃÃ ©ÁôÏžÃ. ©ÂýÃþÃ ¥ÁþÁ¥ÁÅ ¥ÁþÁ program¨Í ©Â™Â¨Ã €ÏýÊ, math.hþÁÅ ¥ÁÅÏžÁÅÂ include úÉ¦Á Âê¨Ã. ƒ readymade ©ÂýÃ¨Í sqrt, log10, sin, cos, tan, pow €þÊ©Ã ÌþÃä. Square root Â¬Â¨ÏýÊ sqrt ©Â™Á©ÁúÁÅÖ. €¨ÂÊ, x þÁÅ yœÍ power úÊ¦Á Â¨ÏýÊ, pow €þÊ žÂþÃþÃ ©Â™Á©ÁúÁÅÖ. x^yþÁÅ pow(x,y) ¨ÂÂ ©Âë¦Á Â¨Ã. €¨ÂÁÊ, log10 €þÊžÃ logarithm þÁÅ ÁþÁÅÍÑ©Á™ÂþÃÃ ©Â™Á©ÁúÁÅÖ.

**Example 11**: Quadratic equation ¦ÉÅÁÑ coefficientsþÁÅ read úÊ¬Ã žÂþÃ descriminant function ¦ÉÅÁÑ square rootþÁÅ print úÊžÂâ¥ÁÅ.

**Solution**:¡Áõ§Ãà program ÃëÏžÁ ‚©Áí£™ÃþÁžÃ. ¥ÁÆ™ÁÅ coefficientsþÁÅ ³ÍÛ§÷ úÉ¦Áê™ÂþÃÃ, result ³ÌÛ§÷ úÉ¦Áê™ÂþÃÃ ¥ÉÅœÁàÏ þÂ¨ÅÁÅ float type variablesþÁÅ ¥ÁÅÏžÁÅÂ declare úÊªÂ¥ÁÅ. œÁ§ÁÅ©ÂœÁ, valuesþÁÅ read úÊ¬Ã, results sqrt žÂí§Â calculate úÊ¬Ã print úÊ¬ÁÅàþÂä¥ÁÅ.

**#include<stdio.h>**

**#include<math.h>**

**int main(){**

**float a,b,c,dis;**

**printf(“Enter a, b, c values:”);**

**scanf(“%f%f%f”, &a, &b, &c);**

**dis = sqrt(b\*b-4\*a\*c);**

**printf(“%f\n”, dis);**

**return(0);**

**}**

ƒ program §ÁÅþ÷ úÊ¬ÁÅàþÁä¡Áôå™ÁÅ, 1 4 1 values ‚¬Êà ¥ÁþÁÁÅ program 3.46þÁÅ print úÊ¬ÁÅàÏžÃ. €žÊ ¥ÁþÁ¥ÁÅ 4 1 1 values input ¨ÂÁ ‚¬Êà, ¥ÁþÁÁÅ Domain Error €þÊ error ©Á¬ÁÅàÏžÃ. ‡ÏžÁÅ©Á¨þÁ?. ƒ values substitute úÊ¬ÃþÁ¡ÁôåžÁÅ b\*b-4\*a\*c value -15 €©ÁôœÁÅÏžÃ. Mathematics¨Í square root ±ÂüÃýÃ©÷ þÁÏ£§ÁìÁÅ ¥Á ÂœÁë¥Ê ©ÁôÏžÃ. ÁþÁÅÁ ¥ÁþÁÁÅ ƒ error ©Á¬ÁÅàÏžÃ. ‚žÃ ÁÅ™Â ‹ run time error.

**Example 12**: ƒ ÃëÏžÁ ‚©Áí£™ÃþÁ program, input ¨ÂÂ amount, rate ¥Á§Ã¦ÁÅ time duration read úÊ¬Ã, ©Á™ÄÝ simple ¥Á§Ã¦ÁÅ úÁÁë ©Á™ÄÝ ¡ÁžÁãœÁÅ¨žÂí§Â calculate úÊ¬ÁÅàÏžÃ. §ÉÏ™ÁÅ ¡ÁžÁâœÁÅ¨¨Í ÁýÃÛþÁ ©Á™ÄÝ¨ ¥ÁŸÁê ¤ÊŸÂþÃä ÁÆ™Á print úÉ¬ÁÅàÏžÃ.

**#include<stdio.h>**

**#include<math.h>**

**int main(){**

**float p, t, r, si, ci, diff;**

**printf(“Enter Principal amount, time and rate\n”) ;**

**scanf(“%f%f%f”, &p, &t, &r) ;**

**si = (p\*t\*r)100 ;**

**ci = p\*pow((1+r/100),t)-p;**

**diff = ci-si ;**

**printf(“Compound Interest=%f\nSimple Interest=%f\nDifference=%f\n”, ci, si, diff);**

**return(0);**

**}**

**2.2.8 The Comma Operator**

§ÉÏ™ÁÅ ¨ÊžÂ ¥ÁÆ™ÁÅ statementsþÁÅ Á¨¡Á™ÂþÃÃ comma operator ©Â™Á©ÁûÁÅÖ. „žÂÿÁ§Á›ÁÅ, ÃëÏžÁ ‚úÃÖþÁ expression ¨Í, First xÁÅ 10 ‚¬ÁÅàþÁä¥ÁÅ, œÁ§ÁíœÁ yÁÅ 5 ‚¬ÁÅàþÂä¥ÁÅ, œÁ§ÂíœÁ x+y, €þÁÂ 15 calculate €©ÁôœÁÅÏžÃ.¥ÉÅœÁàÏ bracket value 15 €©ÁôÏœÁÅÏžÃ, €žÃ value €þÊ variableÁÅ „©Áí£™ÁÅœÁÅÏžÃ.

value = (x = 10, y = 5, x + y);

**2.2.9 Implicit Assignment Operators**

‚©Ã statements ±ÌýÃÛÂ ©Âë¦Á™ÂþÃÃ ©Â™ÁÅœÂ§ÁÅ. „žÂÿÁ§Á›ÁÅ, Salary=Salary+bonus; €þÊ žÂþÃþÃ Salary+=bonus; ¨ÂÁ ©Âë¦Á©ÁúÁÅÖ. ƒ ÃëÏžÁ ¥Á§Ã ÌþÃä examples ©ÁôþÂä¦.

a=a-b; -> a -=b; a=a\*10; -> a\*=10; a=a/b; -> a/=b; a=a%10; -> a%=10;

**2.2.10 sizeof operator**

žÄþÃþÃ ÌÏžÁ§ÁÅ ¢ÁÏÁïþÁÅ €þÃ ÁÆ™Â €ÏýÂ§ÁÅ. žÄþÃþÃ ©Â™ÁÅÌþÃ, ¥ÁþÁ¥ÁÅ ˆ type variableÁÅ ‡þÃä bytes ¥ÁþÁ computer¨Í allocate úÊ¬ÁÅàÏžÍ œÉ¨Å¬ÁÅÍ©ÁúÁÅÖ. ƒ ÃëÏžÁþÁ ‡©Áí£™ÃþÁ ¡Áõ§÷àê program ŠÍÑ variable typeÁÅ ¥ÁþÁ computer¨Í ‡þÃä bytes œÄ¬ÁÅÌÏýÅÏžÍ print úÊ¬ÁÅàÏžÃ.

**Example 13:**

**#include<stdio.h>**

**int main(){**

**int i, j, k, l, m;**

**i = sizeof(char); j = sizeof(int);**

**k = sizeof(long); l = sizeof(float);**

**m = sizeof(double);**

**printf(“%d%d%d%d%d\n”, i, j, k, l, m);**

**return(0);**

**}**

**2.3.11 Operator Precedence in C Language**

Calculations úÊ¬ÁÅàþÁä¡Áô™ÁÅ, ˆ operator ¥ÁÅÏžÁÅ evaluate €©ÁôœÁÅÏžÃ, œÁ§ÂíœÁ ˆžÃ €©ÁôœÁÅÏžÃ, €þÊžÂþÃ ÁÅ§ÁÅÏúÃ ¦ Â ÃëÏžÃ table œÉ¨Ã¦ÁúÊ¬ÁÅàÏžÃ.

|  |  |  |
| --- | --- | --- |
| **Operator Type** | **Operator** | **Associativity** |
| Primary Expression Operators | () [] . -> expr++ expr-- | left-to-right |
| Unary Operators | \* & + - ! ~ ++expr --expr (typecast) sizeof() | right-to-left |
| Binary Operators | \* / % | left-to-right |
| + - |
| >> << |
| < > <= >= |
| == != |
| & |
| ^ |
| | |
| && |
| || |
| Ternary Operator | ?: | right-to-left |
| Assignment Operators | = += -= \*= /= %= >>= <<= &= ^= |= | right-to-left |
| Comma | , | left-to-right |

**2.3 Conclusions**

ము0దుగా ఓ చిన్న ప్రోగ్రామును ఎలా వ్రాయాలో డీటైల్ గా చూశాము. ఈ చాప్టరులో operators గురి0చి కూడా నేర్చుకొన్నాము.