**Data type Purpose Memory Type**

**int to store whole number 2 bytes**

**float decimal upto 6 decimal places 4 bytes**

**double large fractional number 8 bytes**

**char store single character (alphabet, digit & symbol) 1 byte**

**#include<stdio.h>**

**#include<conio.h>**

**Void main()**

**{**

**\Variable declaration**

**\Input to the program**

**\formulation steps or calculation steps**

**\Output of a program**

**getch() \to halt the output**

**}**

**Problem: Develop the algorithm to find the volume of the cylinder?**

**(Pi\*r^2h)**

**Step1: Start**

**Step2: Declare 3 variables (v,r and h)**

**Step3: Input r and h**

**Step4: Calculate volume, v <- 3.14\*r\*r\*h>**

**Step5: Display the volume of cylinder**

**Step6: Stop**

**Output function:**

**printf(“ ”);**

**\n to change line**

**\t for tab spacing**

**Input function:**

**scanf(“format specifier”,&var1, &var2…….&varn);**

**data type format specification**

**int %d**

**float %f**

**double %lf**

**char %c**

**long int**

**Problem: Write a program to calculate average of three numbers**

**Ctrl insert to copy**

**Shift insert to paste**

**Problem: Convert following sentential forms into equivalent conditions**

1. **Number,n is even: if n%2==0 (modulus operator to get reminder)**
2. **Number is positive: if n>0**
3. **Two numbers are not equal: a!=b**
4. **Number is divisible by 5 as well 7: a%5==0 && a%7==0 (II or, ! not)**

**Problem: Classify the roots of quadratic equations and create the equivalent conditions.**

**DAY 2**

**Branching statement/ selection statement/ single pass statement:-**

**If(condition)**

**{**

**//code**

**}**

**Problem: Compute f(x)=1/(1-x), if x<0; f(x)=1/(1+x^2), if x>0; f(x)=0, if x=0.**

**If else**

**Problem: Develope the C program to check whether person is eligible for voting.**

**Problem: To check whether person is eligible for blood donation. Criterion is age between 18 to 55 and weight of the person should be 45 and above.**

**If else ladder**

**Problem: Write a program to write a simple and compound interest.**

**Problem: Company gives a special discount on the purchase of their products on following purchases. Input the purchase amount from the user and calculate the discount and total bill.**

|  |  |
| --- | --- |
| **Purchase amount, p** | **Discount, d** |
| **P<=1000** | **10%** |
| **p>1000 && p<=5000**  **p>5000** | **20%**  **30%** |

**Problem: Write a c program to find the roots of a quadratic equation.**

**Switch**

***switch*(variable){**

***case* constant1:**

**statement**

***break*:**

***case* constant2:**

**statement**

***break*;**

***case* constant3:**

**statement**

***break*;**

***case* constantN:**

**statement**

***break*;**

***default*:**

**statement**

**}**

**Problem: write a to check whether a character is vowel or consonant.**

**switch(ch){**

**case ‘a’:**

**case ‘e’:**

**case ‘i’:**

**case ‘o’:**

**case ‘u’:**

**printf(“Vowel”);**

**break;**

**default:**

**printf(“Consonant”);**

**}**

**Problem: Write the menu driven program to calculate simple calculator.**

**Iterative statements/ looping statements/ multi-pass statements**

**for(exp1 initialisation;exp2 condition;exp3 updation)**

**{**

**}**

**while(condition){**

**\\code**

**}**

**Problem: Write a program to calculate GCD and LCM of any two number.**

**LCM=(a\*b)/GCD;**

**GCD=**

**while(a!=b){**

**if(a>b){**

**a=a-b;**

**}**

**else b=b-a;**

**}**

**do-while**

**do{**

**\\code**

**} while(condition)**

**Problem: Find the sum of individual digits of a given number.**

**Jump statements:**

1. **Break**
2. **Continue**
3. **Return**
4. **Goto label; label:**

**Problem: Write a program to test whether a number is Armstrong number or not.**

**DAY 3**

**ARRAY**

**Int a[n]: memory requirement = n x memory requirement of datatype**

**= n x 2 bytes**

**= 2n bytes**

**Problem: write a program to find the maximum number from the array of numbers.**

**STRING**

**char a[10]**

**Problem: student name, seat number, marks of 3 subjects**

**display total marks and result (all subject above 40)**

**FUNCTION:**

1. **Inbuilt functions**

**Under math.h ex: pow(x,y)=x^y;**

**sqrt(x)=(x)^0.5;**

**exp(x)=e^x;**

**log(x)=log x;**

**ceil(x)=round off to nearest higher whole number;**

**floor(x)=round off to nearest smaller whole number;**

**abs(x)=mod x;**

1. **User defined functions**

**Function declaration:**

**Syntax: returntype functionname(parameter list)**

**Returntype can be void or other than void (int, float, double…….)**

**Function call:**

1. **If returntype is void, then syntax is:**

**functionname(only names of parameter & no datatype);**

1. **If returntype is other than void, then syntax is:**

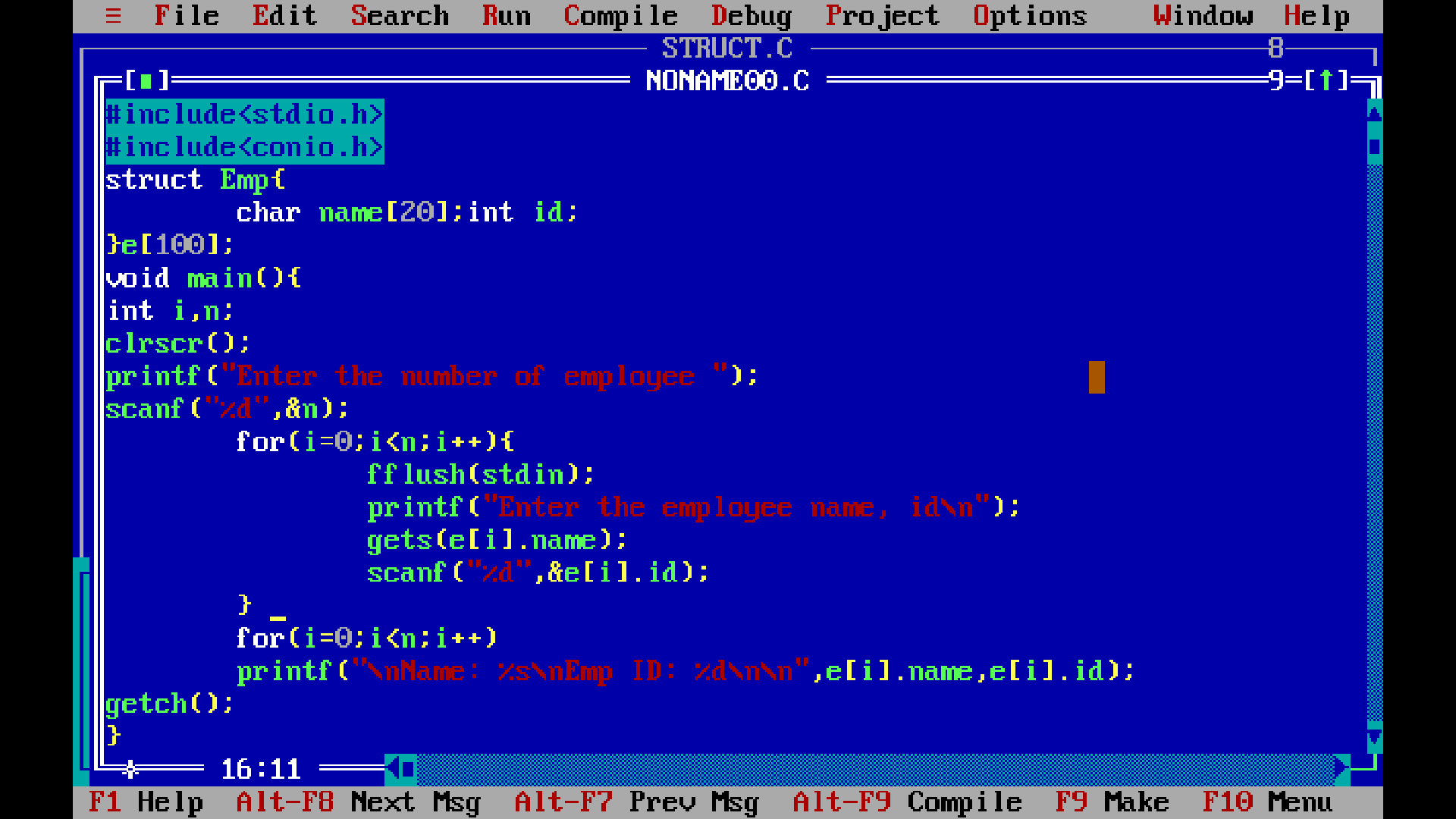
**Resultvariable=functionname(only names of parameter);**

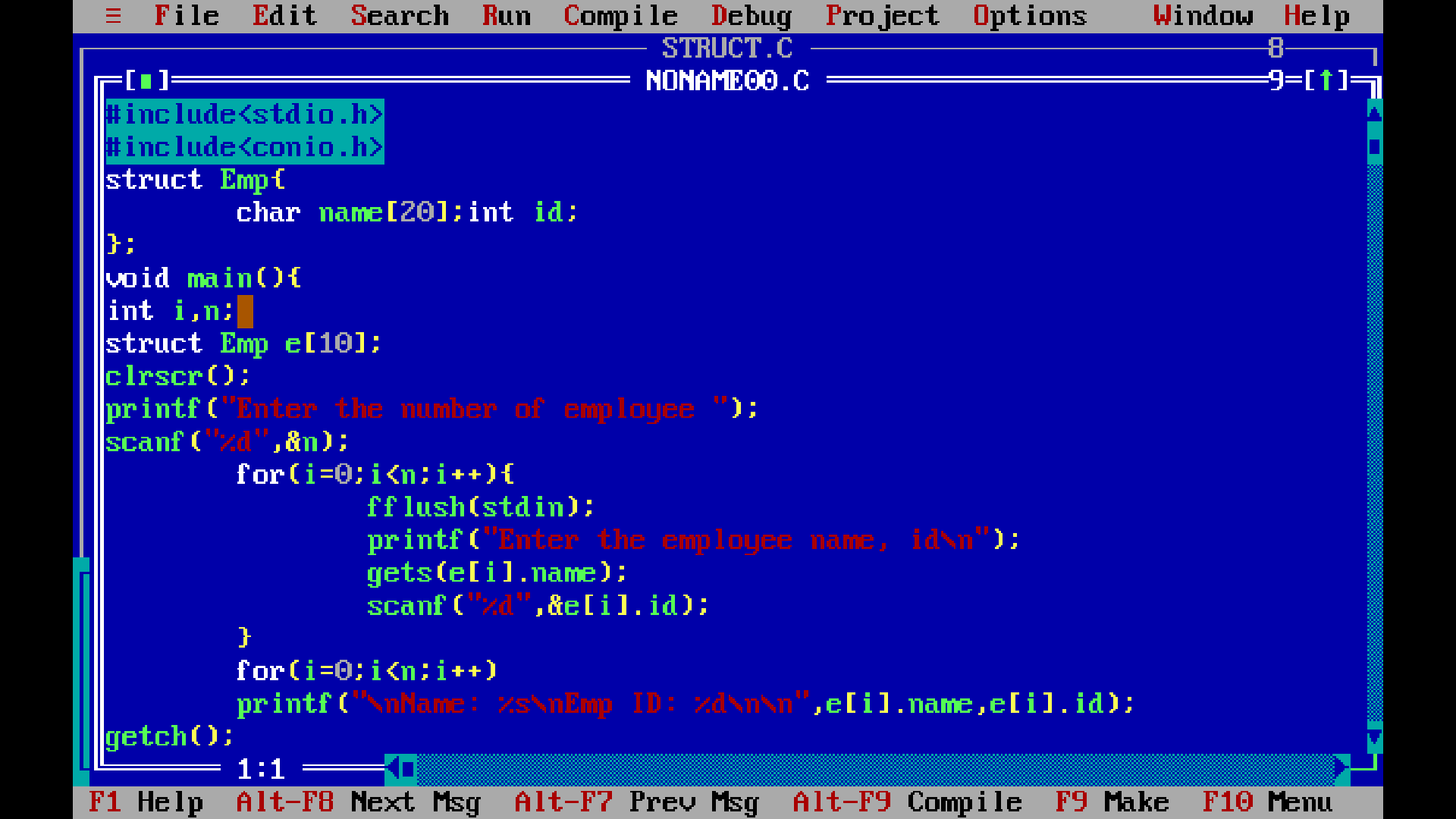
**Problem: Develop a function to calculate factor of a number.**

**DAY 4**

**OBJECTED ORIENTED PROGRAMMING**

**STRUCTURE:**

****

****

**OBJECT ORIENTED PROGRAMING: CLASS CONCEPT**

**There are mainly four OOP principles:**

**Abstraction: act of representing essential features by hiding background task.**

**Encapsulation: wrapping up of data and function in a single unit.**

**Inheritance: to create a new class from the existing class.**

**Polymorphism: using same function to perform many tasks/ ability to take >1 form.**

**class concept:**

**class classroom{**

**private: (no access outside the class, only use from within the class)**

**//members (include variable/function or both)**

**public: (full access outside to the class)**

**//members**

**protected: (can be used by outside by child/derived class)**

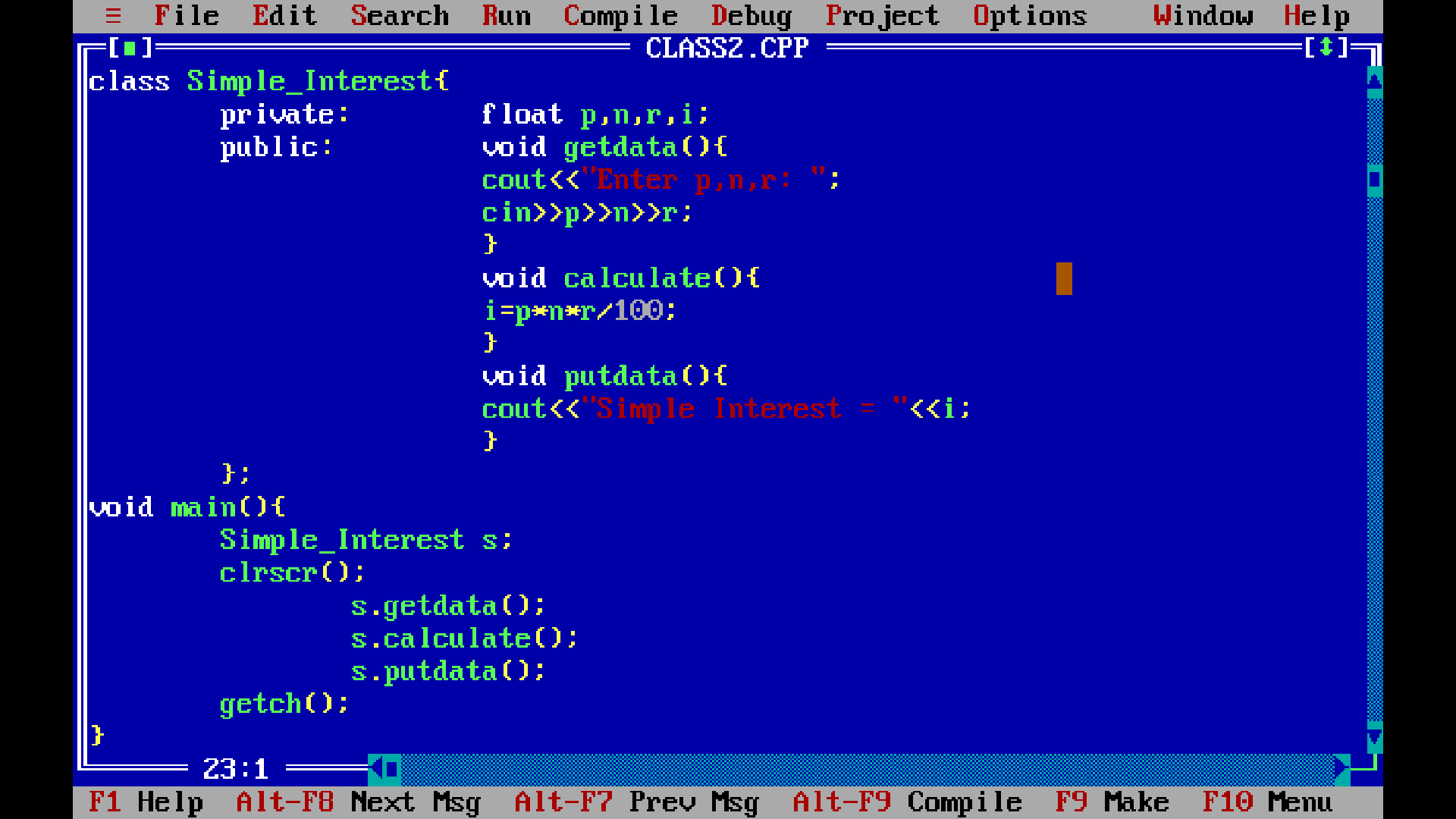
**//members**

**};**

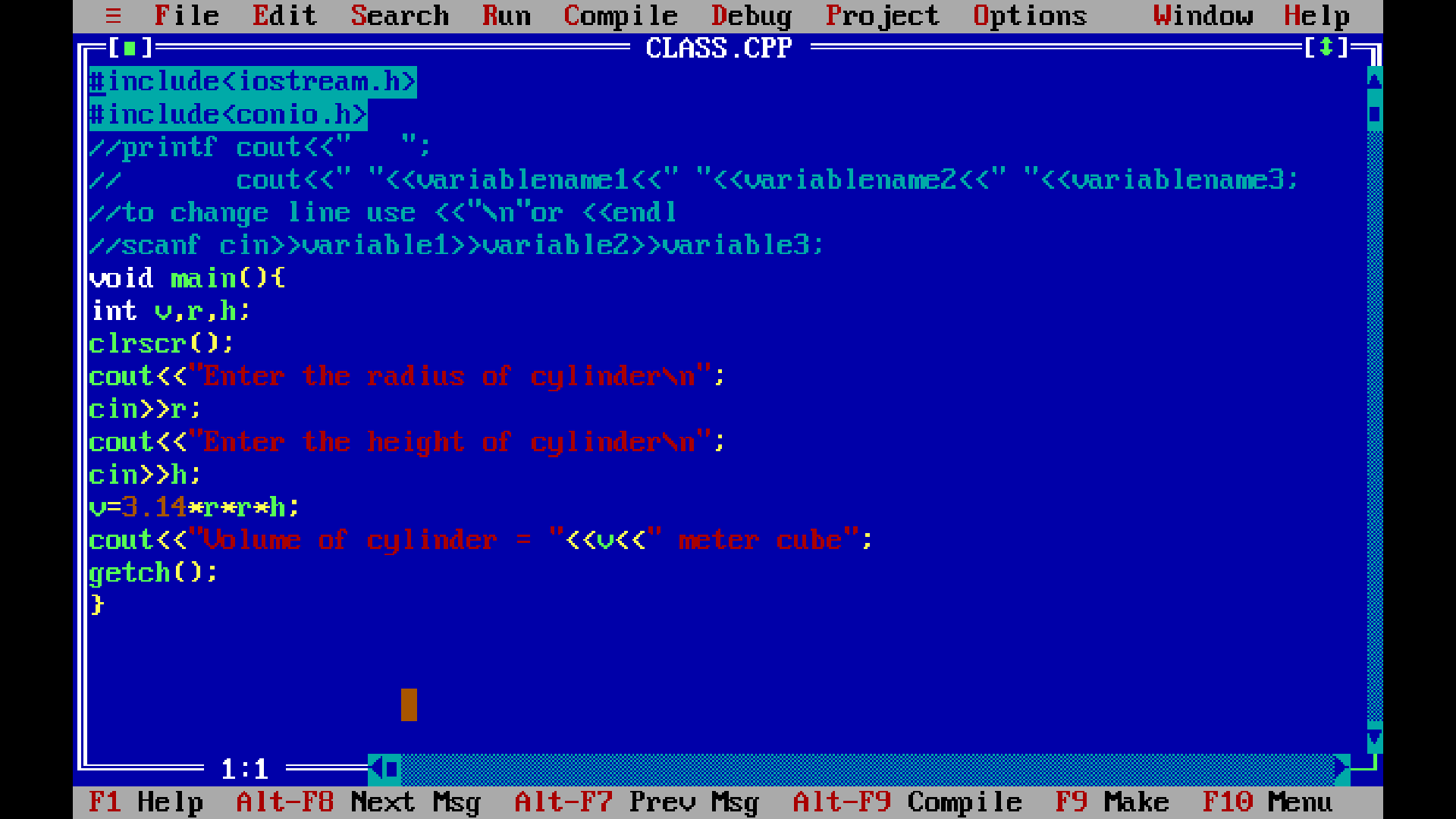
**keywords: class, private, public, protected**

**Problem: Develop the C++ application to calculate simple interest and compound interest. Aplication should have three different functions:**

1. **getdata()-this function should read P,r,t.**
2. **calculate()-to calculate simple interest.**
3. **putdata()-to display simple interest.**

****

**SAMPLE C++ PROGRAM**

****

**Problem: Develop a simple banking application with following functions**

**CreateAccount()**

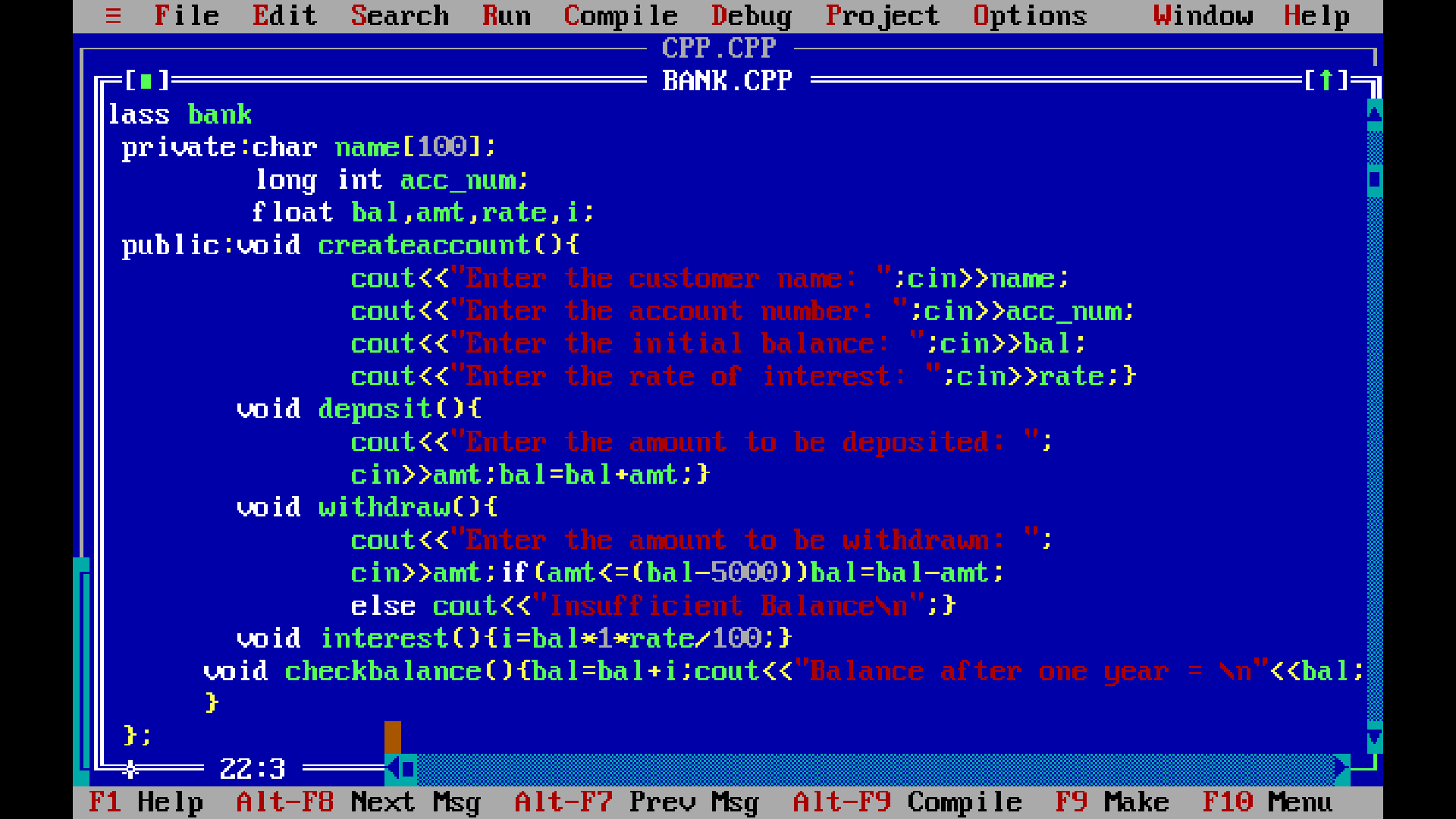
**Deposit()**

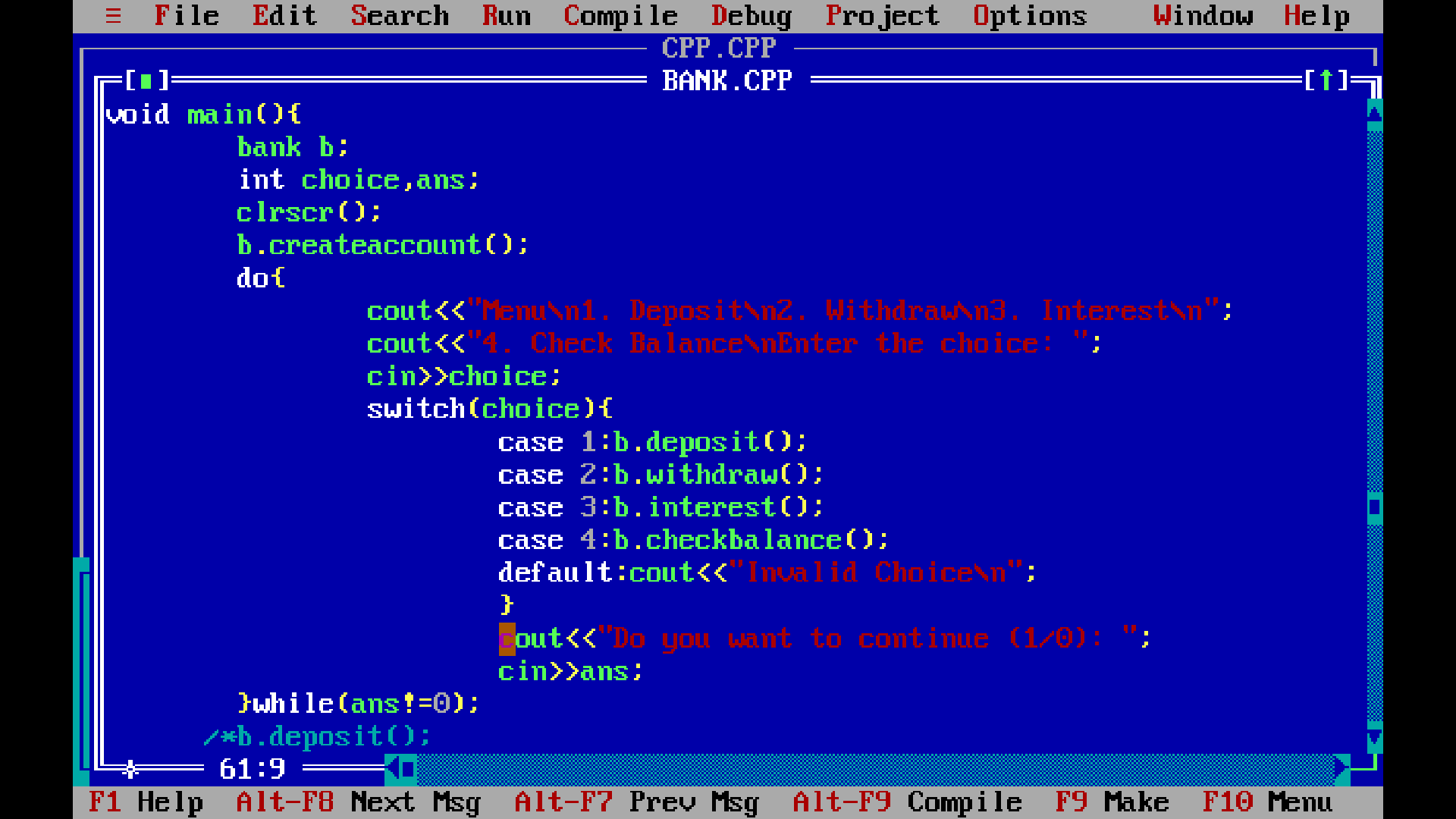
**Withdraw()**

**CalculateInterest()**

**CheckBalance()**

**Application should read the customer name, account number, initial balance, and the rate of interest.**





**Problem: Addition of two complex number.**

**DAY 5**

**CONSTRUCTORS AND DISTRUCTORS:**

**returntype functionname (parameter list){**

**//code**

**}**

|  |  |  |
| --- | --- | --- |
| **Properties** | **Function** | **Constructor** |
| **Name** | **Userdefined** | **Classname** |
| **Returntype** | **Yes** | **No** |
| **Execution** | **Manual (on call)** | **Automatic (during object creation)** |
| **Type** | **Static or non-static** | **Non-static** |
| **Categories of the function** | **Not categorised based on the parameter** |  |
|  |  |  |

**class Test{**

**private:**

**int a;**

**float b;**

**static int c;**

**public:**

**void abc(){**

**//code**

**}**

**static void pqr(){**

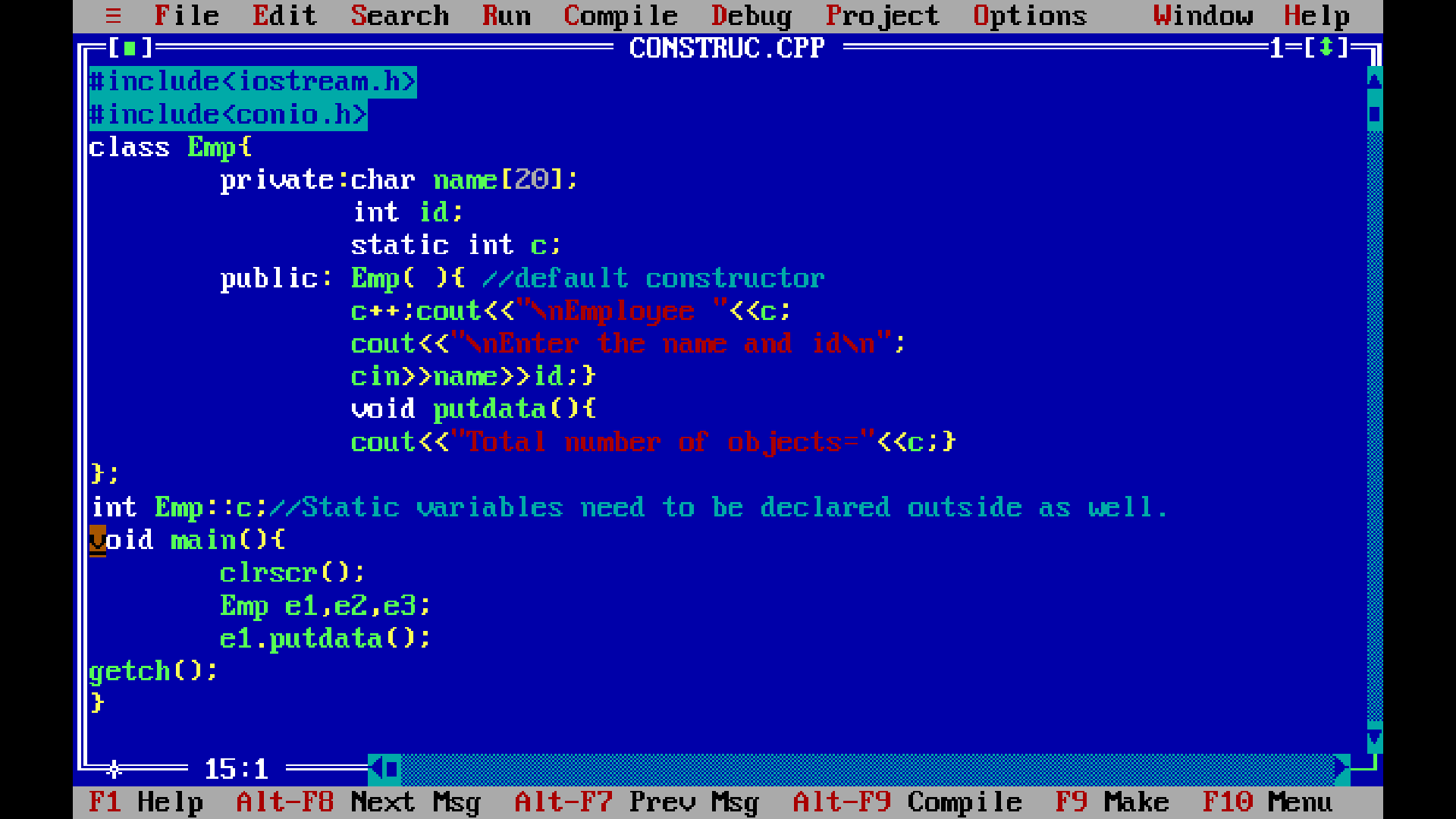
**//code**

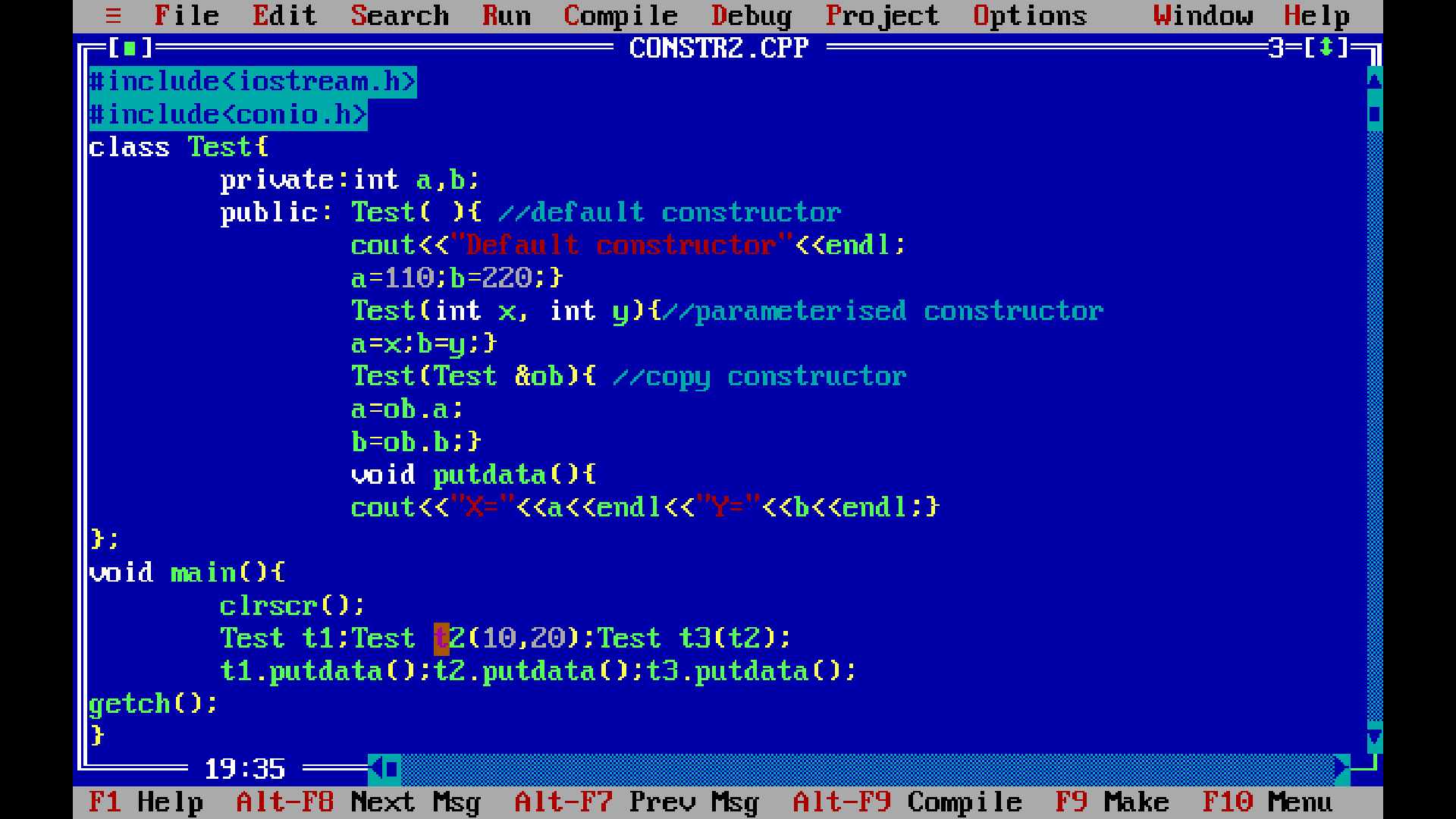
**}**

**}**

**static member access- classname:: staticmembername**

**Constructor example:**





**INHERITENCE:**

1. **Concept of inheritence.**
2. **Types of inheritence:**
3. **Single inheritence (1:1)**
4. **Multi-level inheritence (1:1:1:1:…….:1)**
5. **Multiple inheritence (many:1)**
6. **Hierarchial inheritence (1:many)**
7. **Hybrid inheritence (mixed inheritence)**
8. **Syntax to inherit class**

**class derived-class-name : access-mode base-class-name**

**{**

**// body of class**

**};**

**Example:**

**class Emp**

**{**

**// code**

**};**

**class Manager : public Emp**

**{**

**// code**

**};**

**class Scientist : public Emp**

**{**

**// code**

**};**

**Problem: Implement the following inheritence**

**Emp**

**Name**

Getdata() putdata()

**Id**

**salary**

**Manager Scientist**

**Projects, teamsize patents, publications**

**………………………………………………………………………………………………………………………...**

**#include<iostream.h>**

**#include<conio.h>**

**#include<stdio.h>**

**class Emp{**

**private:char name[20];int id;float salary;**

**public: void getdata(){**

**fflush(stdin);**

**cout<<"Enter the name, id and salary\n";**

**cin>>name>>id>>salary;}**

**void putdata(){**

**fflush(stdin);**

**cout<<"\n\nName="<<name<<"\nID="<<id<<"\nSalary="<<salary<<endl;}**

**};**

**class Manager : public Emp{**

**private:int teamsize, projects;**

**public: void getdata(){**

**fflush(stdin);**

**Emp::getdata();**

**cout<<"\nEnter teamsize and number of projects handled\n";**

**cin>>teamsize>>projects;}**

**void putdata(){**

**fflush(stdin);**

**Emp putdata();**

**cout<<"TeamSize="<<teamsize<<"\nNumber of projects handled="<<projects<<endl;}**

**};**

**class Scientist : public Emp**

**{**

**private:int pub, patents;**

**public: void getdata(){**

**fflush(stdin);**

**Emp::getdata();**

**cout<<"\nEnter the number of Publications and Patents";**

**cin>>pub>>patents;}**

**void putdata(){**

**fflush(stdin);**

**Emp putdata();**

**cout<<"Publications="<<pub<<"\nPatents="<<patents<<endl;}**

**};**

**void main(){**

**Manager m;**

**Scientist s;**

**clrscr();**

**cout<<"Enter the manager info\n";**

**m.getdata();**

**cout<<"\nManager info is:\n";**

**m.putdata();**

**cout<<"Enter the Scientist info\n";**

**s.getdata();**

**cout<<"\nScientist info is:\n";**

**s.putdata();**

**getch();**

**}**

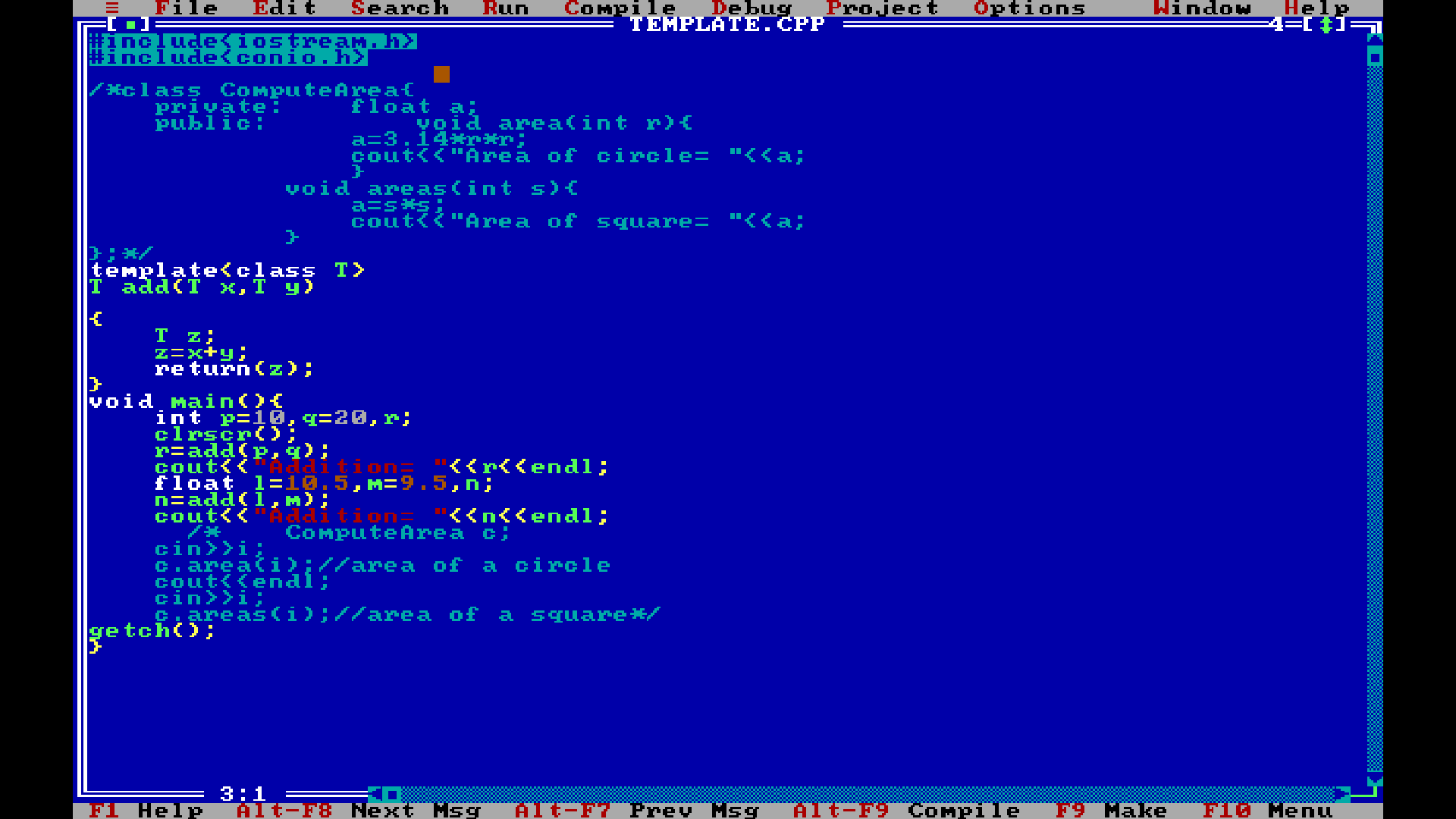


**TEMPLATE:**

**This allows a function or class to work on many different data types without being ……..PPT**

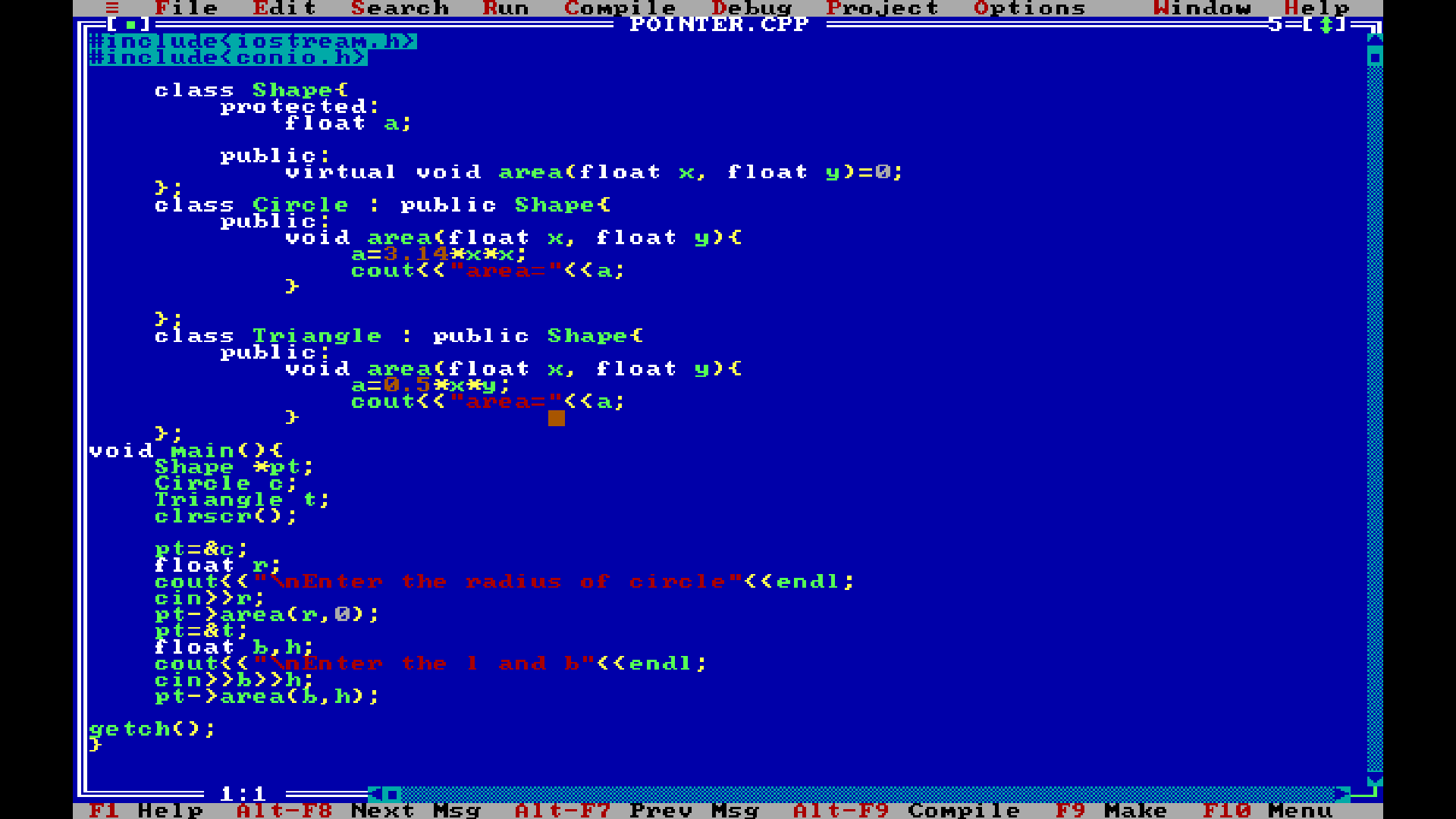
**syntax: template<class Ttype>**

**example below:**



**DAY 6**

**VIRTUAL FUNCTION: Example:-**



**EXCEPTION HANDLING:**

**Runtime error at any code and syntax error (try, throw, catch)**

**FILE HANDLING:**