WEEK 3

AASMA ABDUL WAHEED (FOIT & CS)

سورة طه پارہ: 16

قَالَ رَبِّ اشْرَحْ لِيْ صَدْرِيْ ۙ ﴿۲۵﴾

[قَالَ: اس نے کہا] [رَبِّ: اے رب !] [اشْرَحْ: کھول دے] [لِيْ: میرے لیے] [صَدْرِيْ : میرا سینہ]

وَيَسِّرْ لِيْٓ اَمْرِيْ ۙ ﴿۲۶﴾

[وَيَسِّرْ: اور آسان کر دے] [لِيْ: میرے لیے] [اَمْرِيْ : میرا کام]

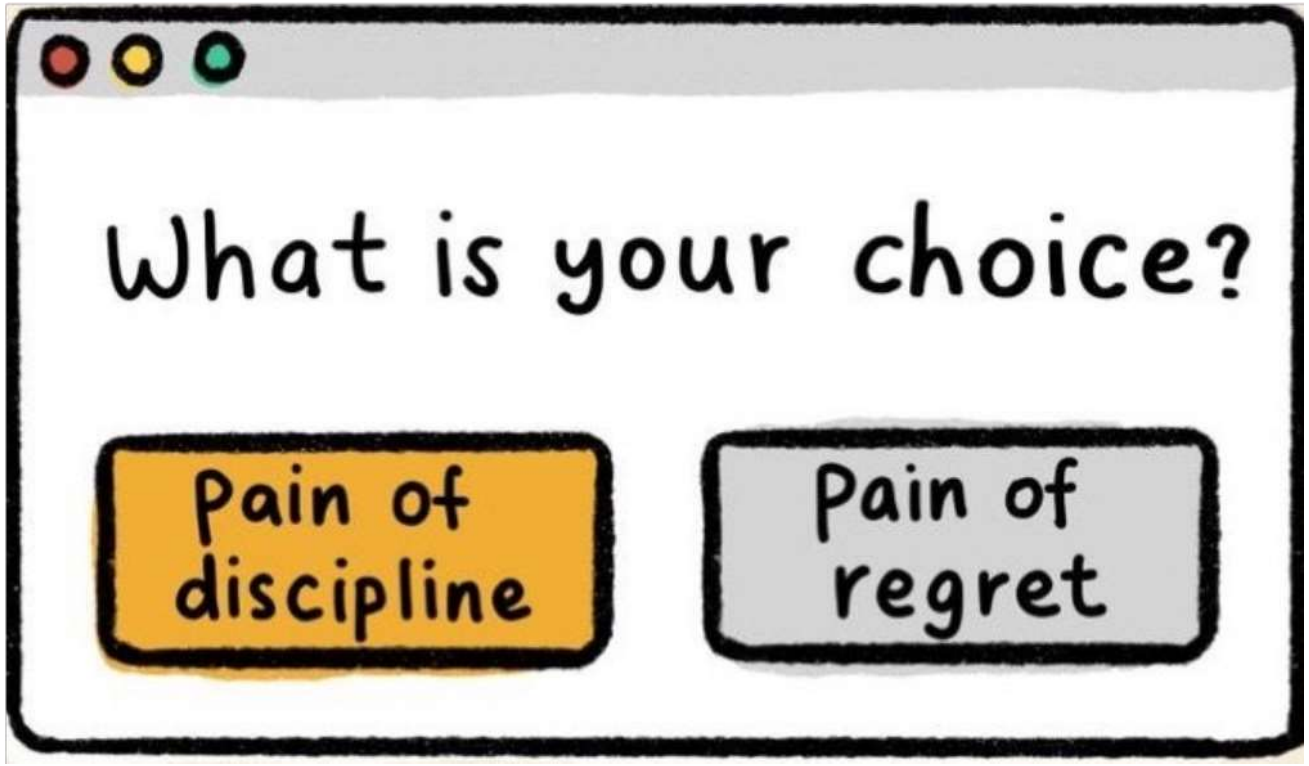
وَاحْلُلْ عُقْدَةً مِّنْ لِّسَانِيْ ۙ ﴿۲۷﴾

[وَاحْلَلْ: اور کھول دے] [عُقْدَةً: گرہ] [مِّنْ: سے] [لِّسَانِيْ: میری زبان]

يَفْقَهُوْا قَوْلِيْ ۫ ﴿۲۸﴾

[يَفْقَهُوْا: وہ سمجھ سکیں] [قَوْلِيْ : میری بات]

QUESTIONS / FEEDBACK / CONCERNS


What is your choice?
Pain of discipline
Pain of regret

EVALUATING YOUR GROWTH

RECAP MODULE 2

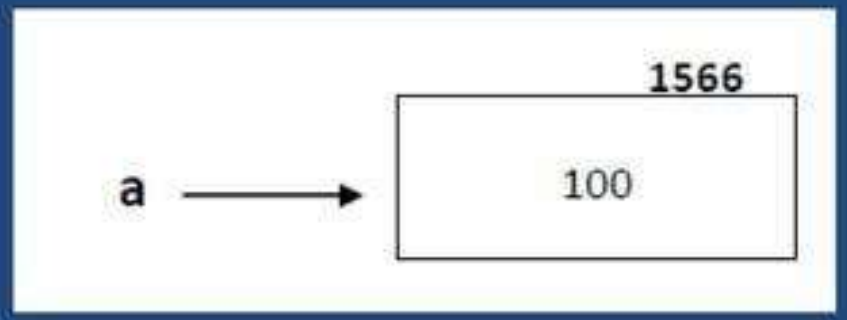
- **Hello World Program in C++**
  - Processing a C++ program/Execution Flow
  - Syntax of C++ (cout << “literal string \n”)
  - Syntax of C++ (cout << Numeric Constant/Expression)
  - Comments/ Importance of Comments
  - Syntax Errors
  - Syntax vs. Semantics
- **Arithmetic expression**
  - Output Numbers (Literal Constants) (cout << 2 << endl;)
  - Arithmetic Operators (+, -, *, /, %)
  - Defining Expression/Arithmetic Expression
  - Operator Precedence & Associativity
  - Arithmetic Expression evaluation
  - Output value of an Arithmetic Expression (cout << 2*3 << endl;)
- **Problem Solving using Arithmetic Expression (literal constants)**

MODULE 3

- **Variable**
  - Concept/Purpose of variable
  - Concept of Type
  - Declaration of variable/Syntax, Semantics
  - The rule for a valid identifier
  - Initialization of variables
  - Accessing values of a variable
  - Assignment statement, Syntax/Semantics
  - Arithmetic Expression involving variables and/or literal constants
  - **Problem-solving using literal constants, variables**
- **Interactive Program (Taking Input from Console)**
  - How to take input from the user.
  - Syntax/Semantics of cin (Formatted Input)
  - Concept of white space (space, tab, newline) and default working of cin
  - **Problem Solving using cin, variables, strings**

Variables

- Variable is simply name given to memory location which acts as placeholder or container for storing data. It may help to think of variables as a container for a value.



What is a data type?

- When we wish to store data in a C++ program, we have to tell the compiler which type of data we want to store.
- The data type will have characteristics such as:
  - The range of values that can be stored.
  - and the operations that can be performed on variables of that type.

Data type

- We can store variables in our computer memory. Since each type of data takes different amount of memory, like integer takes 2 bytes, decimal numbers take 4 byte, and SINGLE character takes 1 byte. So we have to tell computer about the type of data we are going to store in our variable

C++ Built-in Data Types

- Called fundamental types or primitives types:
  - Numerical (integer and floating point)
  - Character
  - Logical (Boolean)

`int` Data Type

- Integer variables represent integer numbers like 1, 30,000 and -45.
- Integer variables do not store real numbers.

Integer Variables Example

```
#include<iostream>
int main ( )
{
  int var1;                    //define var1
  int var2;                    //define var2
  var1  =  20;                 //assign value to var1
  var2 = var1 + 10;                  //assign value to var2
  std::cout<<“Result =”;
  std:: cout<<var2<< endl; //displaying the sum of
  var1+10
  return 0;
}
```

Floating-Point Types

- Floating point types can contain decimal numbers.
  - Examples: 1.23, -.087.
- There are three sizes:
  - float (single-precision)
  - double (double-precision)
  - and long double (extended-precision).
- Examples:

```
float Temp= 37.623;
double fahrenheit = 98.415;
long double accountBalance = 1897.23;
```

Example

```
#include<iostream.>
int main ( )
{
  float  rad, area;
  float   PI =3.14;
  std::cout << “Enter radius of circle”;
  std::cin >> rad;
  area = PI * rad * rad;
  std::cout <<“Area is” << area << std::endl;
  return 0;
}
```

char Data Type

- Used for characters: letters, digits, and special symbols.
- Each character is enclosed in single quotes.
- Some of the values belonging to char data type are:

  'A','a','0','*','+','$','&'.
- A blank space is a character and is written ' ', with a space left between the single quotes.

Examble

```
include<iostream>
int main ( )
{
 char  charvar1 = 'A';     //define char variable
 char  charvar2  = '\t';
  std::cout <<  charvar1;          // display character
std::cout   << charvar2;
 charvar1 = 'B';              //reassigning charvar1 to B
  std::cout   << charvar1;
 return (0);
}
```

bool Data type

- Has two values (true) and (false).
- Manipulate logical expressions.
- true and false are called logical values.
- bool, ture, and fasle are reserved words.
- For example:

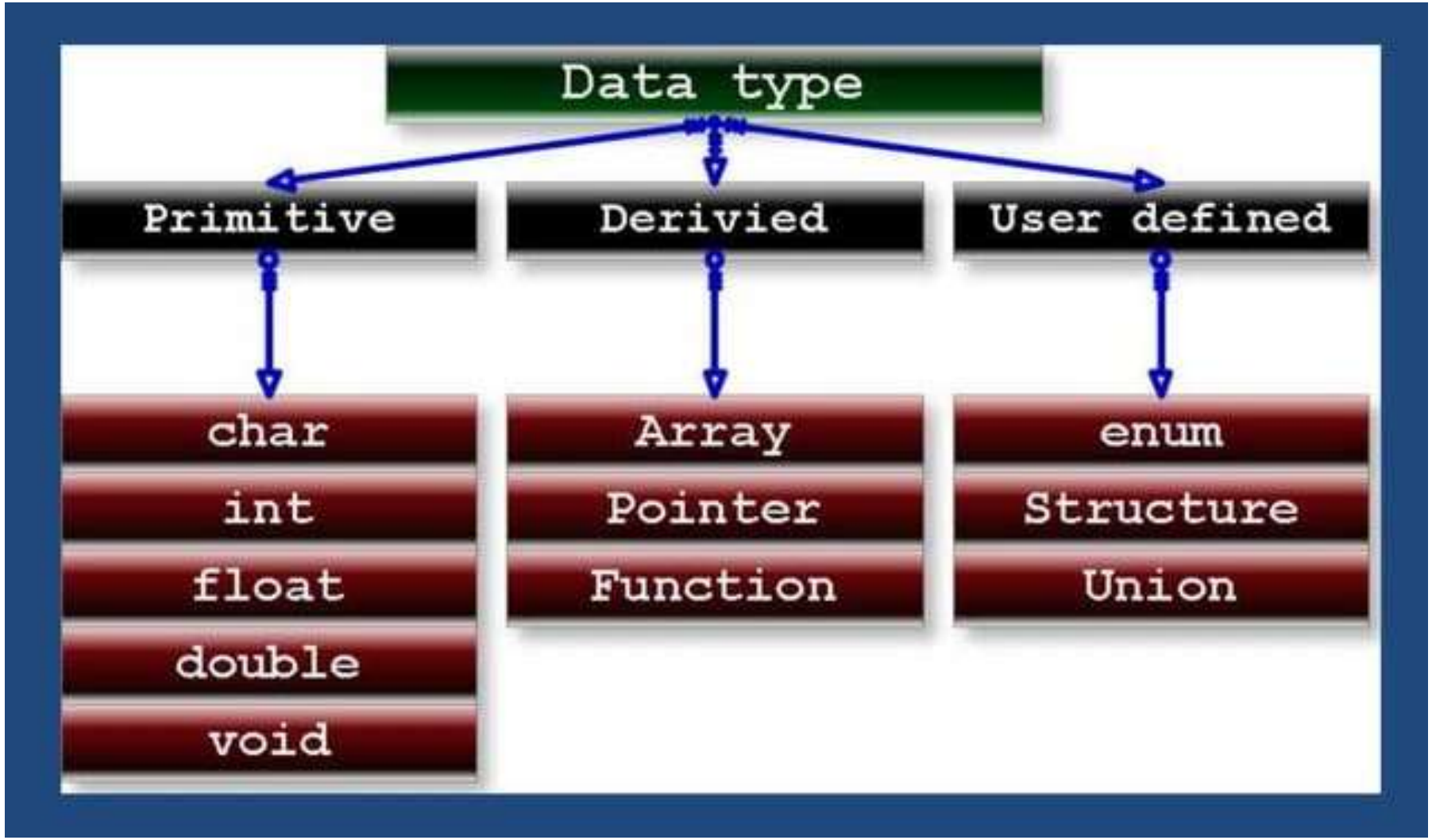
```
bool isEven = false;
bool keyFound = true;
```

DATA TYPES

- **Predefined/inbuilt** :
  These datatypes are inbuilt.
  eg : char, int, float, double

- **Drived Datatype**
  These datatypes are inherited from predefined datatype. eg : arrays, pointer.

- **Userdefined datatype** : These are created by user for his/her own purpose
- eg : typedef, enum, structure, union


Data type
Primitive
Derivied
User defined
char
int
float
double
void
Array
Pointer
Function
enum
Structure
Union

Predefined Datatypes

Type	Keyword
Boolean	Bool
Character	Char
Integer	Int
Floating point	Float
Double floating point	Double
Valueless	Void
Wide character	wchar_t

Variable Type	Keyword	Bytes Required	Range	Format
Character (signed)	Char	1	-128 to +127	%c
Integer (signed)	Int	2	-32768 to +32767	%d
Float (signed)	Float	4	-3.4e38 to +3.4e38	%f
Double	Double	8	-1.7e308 to + 1.7e308	%lf
Long integer (signed)	Long	4	2,147,483,648 to 2,147,438,647	%ld
Character (unsigned)	Unsigned char	1	0 to 255	%c
Integer (unsigned)	Unsigned int	2	0 to 65535	%u
Unsigned long integer	unsigned long	4	0 to 4,294,967,295	%lu
Long double	Long double	10	-1.7e932 to +1.7e932	%Lf

Declaration and initialization of variable

- **How to declare variable?**
- Declaration of variable means specify data type of variable and name assigned **(identifier).**
- A variable declaration provides assurance to the compiler that there is one variable existing with the given type and name so that compiler proceed for further compilation without needing complete detail about the variable.

Variable declaration

```
Data type VarName;
```

- All variables must be declared anywhere in program with a name and data type before they used.
- Begin with a data type then variable name.
- Variables of the same type can be declared in
  - Multiple lines
  - One line separated by commas.

```
int num1;
int num2;
int num3;
```

```
int num1,num2,num3;
```

DECLARING A VARIABLE

- syntax :

  datatype varible_name;

eg :

```
int a;
float b;
char c;
double first;
int i, j, k;
```

Initialization

- It means assigning the values to variable

```
int a;          float b;          char c;
a = 10;         b = 30.05;        c = 'a';
```

Declaration and initialization in single step

```
int  a = 10;

float b = 30.05;

char c = 'a';
```

(Value to characters always assigned in single quotes)

Assigning initial value to variable

Three way to do that:

1- in the declaration of the variable.

Ex:

```
char w='A';
int sum=0;
```

2- Using assign statement.

Ex:

```
i=1;
w='A';
```

3- Accepting the value from user using cin function.

Ex:

```
cin>>number;
```

Variables

Variable:

- Location on computer's memory to store data then use and change its value in a program.

Name(Identifier)

- You can use upper and lowercase letters, digits from 0 to 9, and underscore(_).
- The first character must be letter or underscore (_).
- Don't contains any blank,<,>,&,!,%,^,*,…etc.
- C++ is case sensitive –uppercase and lowercase letters are different, so a1 and A1 are different identifier.
- Not a keyword(int, float, double char, void, return main)
- Meaningful
- Don't give the same name to two different variables.

Type:

- Programmer defined
- Built-in

Identifiers

- Identifier is simply a name given to your variable, class, union, structure, function, etc.
- **Rules for creating valid identifiers:**
- 1. An identifier can be any combination of alphabets, digits, underscore.
- 2. Neither spaces nor special symbol other then underscore can be used.
- 3. It won‘t begin with a digit

Valid Identifiers	Invalid Identifiers
A_	9a
_a	A_B
A56	5a6
Hello	Hello?
Hello_hi	Hello hi
A9b	A9b$

Exercises

Ex1:

Time, TIME, time

Are three correct names for three different identifier?

Ex2:Which of the following is a correct identifier:

_num	Correct identifier
DD-X	Incorrect, the symbol –cannot be used in identifier
12abc	Incorrect, an identifier cannot begin with digits
Name55	Correct identifier
return	Incorrect , keyword
A1	Correct identifier
Name1/4	Incorrect, the symbol /cannot be used in identifier

Keywords

- Are simply the reserved words whose meaning has been explained in library of C language.
- There are 32 keywords in c language.
- **We can't use keywords as identifiers or variables** because if we do so we are trying to assign a new meaning to the keyword, which is not allowed by the computer

List of Keywords

Auto	Goto	Struct	For
double	sizeof	break	signed
int	volatile	else	void
union	do	long	default
const	if	switch	extern
float	static	case	return
short	continue	enum	typedef
unsigned	while	register	char

```
int   float;
char   if;
Int   void;
```

Above will give **error** as we are using keywords as identifier or variable name

Constants

- Is an identity which does not change its value. Once the value is initialized, that can't be changed.
- Constants are just like variables only condition is that their value doesn't change.
- We will use **const** keyword for declaring constants

Syntax:

```
const datatype        varible_name = value;
      const int a = 10;
```

Memory Concepts

- Variable names such as : number1, number2 and sum correspond to locations in the computer's memory.
- Every variable has four parts:
  - Type, name, size and value.
  - Example:
  - char letter='A';
    - Type? Name? Size? Value?

Scope Of Variable

- The scope of variable is the portion of the program where the variable can be used.
- Scope can be:
  - Local
  - Global
- Local variables:
  - Defined within a module
  - Can be seen and used only by the module itself
  - Store temporally in memory
  - Erased when he module terminates

Scope Of Variable

- Global variables:
  - Defined outside any module.
  - Used an seen by all modules
- Variable name can be duplicated within and outside a modules
  - Differentiate between them by using unary scope resolution operator (::)

Examples:

```
int main()
{
  int i;
  char a;
}
```

i: Local variable

```
int i;
int main()
{
  char a;
}
```

i: Global variable

Unary Scope Resolution Operator

- Denoted as (::)
- Used to declare local and global variables have a same name.
  - To avoid conflicts.
- Syntax: :: variable
  - Example: y= ::x+3
- Not needed if names are different

Example

```
#include <iostream>
using namespace std;

int count = 100;
int main()
{
   int count = 10;
   int Second_count = 50;
   cout << "Local  count = " << count << endl;
   cout << "Global count = " << ::count << endl;
   cout << "Local Second count = " << Second_count <<endl;

   return 0;
}
```

ACCESSING VARIABLE VALUES

- Reading Values: Use in expressions and output
- Example:

```
int age = 20;
cout << age;  // Displays 20
```

ARITHMETIC EXPRESSIONS

- Basic operators +, -, *, /, %
- Using variables and constants
- Example:

  - int a = 5, b = 10;
  - int sum = a + b;.

PROBLEM SOLVING WITH VARIABLES AND CONSTANTS

- **Example:** Temperature conversion (C = (F - 32) * 5 / 9)
- Constants (5, 9, 32) and variables (F, C) used together.
- Explanation: Variables store input, constants remain unchanged

INTERACTIVE PROGRAMS

- Programs that take input from users
- Example: input name and age, output message.

```
string name;
int age;
cout << "Enter your name: ";
cin >> name;
cout << "Enter your age: ";
cin >> age;
```

TAKING INPUT FROM CONSOLE

- Using cin
- cin>> variable; //reads input into a variable
- Example:

```
int age;
cout << "Enter age: ";
cin >> age;
```

CONCEPT OF WHITESPACE IN CIN

- Whitespace characters (space, tab, newline)
- Default cin behaviour is that it ignores leading whitespace and reads until next whitespace
- Example:

```
int num1, num2;
cout << "Enter two numbers: ";
cin >> num1 >> num2;
```

USING \N AND \T FOR FORMATTING

```
#include <iostream>
using namespace std;

int main() {
    cout << "Using \\n for newline:" << endl;
    cout << "Hello\nWorld\nC++ Programming\n" <<
endl;

    cout << "Using \\t for tab spacing:" << endl;
    cout << "Name\tAge\tGrade" << endl;
    cout << "Alice\t20\tA" << endl;
    cout << "Bob\t21\tB" << endl;
    cout << "Charlie\t19\tA" << endl;

    return 0;
}
```

```
Using \n for newline:
Hello
World
C++ Programming

Using \t for tab spacing:
Name    Age     Grade
Alice   20      A
Bob     21      B
Charlie 19      A
```

PROBLEM SOLVING USING CIN

- Example: Sum of two numbers

```
int a, b;
cout << "Enter two numbers: ";
cin >> a >> b;
int sum = a + b;
cout << "Sum is: " << sum;
```

ACTIVITIES

- Swap two numbers by using third variable and without using third variable (Human Variables)
- Take input for a name, age, and favorite hobby. Then, display a customized greeting that includes all three.
- Prepare budget for weekend activities including expenses like (food, transport, entertainment)
- Identifier Relay Race (2ndPlace, _total, result!, finalSum which is correct?)
- Interactive Story Creation

SWAP TWO NUMBERS BY USING & WITHOUT USING THIRD VARIABLE

```
#include <iostream>
using namespace std;

int main() {
    int a = 5, b = 10;

    // Using a third variable
    int temp = a;
    a = b;
    b = temp;
    cout << "After swap (using third variable): a = " << a << ", b = " << b << endl;

    // Without using a third variable
    a = 5;
    b = 10;
    a = a + b;
    b = a - b;
    a = a - b;
    cout << "After swap (without third variable): a = " << a << ", b = " << b << endl;

    return 0;
}
```

TAKE INPUT FOR A NAME, AGE, AND FAVORITE HOBBY. THEN, DISPLAY A CUSTOMIZED GREETING THAT INCLUDES ALL THREE.

```
#include <iostream>
#include <string>
using namespace std;

int main() {
    string name, hobby;
    int age;

    cout << "Enter your name: ";
    getline(cin, name);
    cout << "Enter your age: ";
    cin >> age;
    cin.ignore(); // To ignore the newline character after age input
    cout << "Enter your favorite hobby: ";
    getline(cin, hobby);

    cout << "Hello, " << name << "! You are " << age << " years old and enjoy " << hobby <<
"." << endl;

    return 0;
}
```

PREPARE BUDGET FOR WEEKEND ACTIVITIES INCLUDING EXPENSES LIKE (FOOD, TRANSPORT, ENTERTAINMENT)

```
#include <iostream>
using namespace std;

int main() {
    double food, transport, entertainment, total;

    cout << "Enter your budget for food: ";
    cin >> food;
    cout << "Enter your budget for transport: ";
    cin >> transport;
    cout << "Enter your budget for entertainment: ";
    cin >> entertainment;

    total = food + transport + entertainment;
    cout << "Your total weekend budget is: $" << total << endl;

    return 0;
}
```

IDENTIFIER RELAY RACE (2NDPLACE, _TOTAL, RESULT!, FINALSUM WHICH IS CORRECT?)

```
#include <iostream>
using namespace std;

int main() {
    // Correct identifiers
    int _total = 100;
    int finalSum = 200;

    // Incorrect identifiers
    // int 2ndPlace; // Can't start with a number
    // int result!; // Can't contain special characters other than underscore (_)

    cout << "_total: " << _total << endl;
    cout << "finalSum: " << finalSum << endl;

    return 0;
}
```

INTERACTIVE STORY CREATION

```
#include <iostream>
#include <string>
using namespace std;

int main() {
    string characterName, place, action;

    cout << "Enter a character name: ";
    getline(cin, characterName);
    cout << "Enter a place: ";
    getline(cin, place);
    cout << "Enter an action: ";
    getline(cin, action);

    cout << characterName << " went to " << place << " and decided to " << action << "." <<
endl;
    cout << "What happens next? That's for you to decide!" << endl;

    return 0;
}
```

THANKYOU