فصل دوم

متغیرها و انواع داده ای

در فصل قبل ما چندین خط کدنویسی کردیم تا تنها بتوانیم یک جمله را در صفحه نمایش نشان دهیم در صورتی که ما می توانستیم آن را براحتی در صفحه نمایش تایپ کنیم. اما برنامه نویسی به چاپ کردن متن در صفحه نمایش خلاصه نمی شود. برای اینکه بتوانیم برنامه های کارآمد و مفید بنویسیم نیاز است تا با مفهوم متغیر آشنا شویم.

فرض کنید من از شما می خواهم عدد ۵ را در ذهن خود به خاطر بسپارید. سپس می گویم عدد دو را حفظ کنید. حالا می گویم به عدد اول یک واحد اضافه کنید. آنوقت از شما می خواهم عدد دوم را از عدد اول کم کنید و نتیجه را به من بگویید. نتیجه ۴ می شود. این عملیاتی که شما در ذهن خود انجام دادید تشبیهی است از عملیاتی که کامپیوتر برای همین کار انجام می دهد. کد زیر همان کاری را که شما در ذهن خود انجام دادید، محاسبه می کند:

```
a = ٥;
b = ٢;
a = a + ١;
result = a - b;
```

این نمونه ای از یک عملیات ساده بر روی اعداد صحیح بود. کامپیوتر قادر است میلیونها عدد را همزمان در حافظه خود ذخیره کند و روی آنها عملیات پیچیده ریاضی انجام دهد.

برای این منظور ما می توانیم یک متغیر جهت اختصاص بخشی از حافظه تعریف کنیم و عددی را در آن ذخیره کنیم. هر متغیر نیاز به یک نام یا معرف دارد تا از دیگر متغیرها قابل تفکیک باشد. به عنوان مثال در کد قبل ما سه متغیر با نام های a ، b و result داشتیم به همین ترتیب ما می توانیم هر نامی برای متغیر خود انتخاب کنیم البته تا زمانی که قواعد نامگذاری را رعایت کنیم.

شناسه

یک شناسه (نام متغیر) معتبر رشته ای از یک یا چند حرف، رقم و یا زیر خط (_) می باشد. فضای خالی، نشانه ها و کاراکترهای نقطه گذاری نمی توانند در نام یک متغیر به کار روند. نام یک متغیر باید با حرف و یا زیر خط شروع شود. نکته دیگر هنگام انتخاب نام این است که نباید از کلمات رزرو شده توسط کامپایلر استفاده کرد زیرا باعث تداخل شده و ایجاد مشکل در اجرای برنامه می کند. کلمات رزرو شده عبارتند از:

```
asm, auto, bool, break, case, catch, char, class, const, const_cast,
continue, default, delete, do, double, dynamic_cast, else, enum,
explicit, export, extern, false, float, for, friend, goto, if, inline,
```

```
int, long, mutable, namespace, new, operator, private, protected,
public, register, reinterpret_cast, return, short, signed, sizeof,
static, static_cast, struct, switch, template, this, throw, true, try,
typedef, typeid, typename, union, unsigned, using, virtual, void,
volatile, wchar_t, while
and, and_eq, bitand, bitor, compl, not, not_eq, or, or_eq, xor, xor_eq
```

هر کامپایلر ممکن است علاوه بر کلمات ذکر شده کلمات دیگری را نیز رزرو کرده باشد.

- ✓ زبان C++ یک زبان حساس به حالت حروف می باشد (case sensitive) به این معنی که حالت کوچک یک حرف با حالت بزرگ آن حرف متفاوت است. به عنوان مثال Result ، RESULT و result با یکدیگر متفاوت بوده و هر کدام می توانند یک متغیر جداگانه باشند.

انواع داده ای بنیادی

هنگام برنامه نویسی ما متغیرها را در حافظه کامپیوتر ذخیره می کنیم. ولی قبل از آن باید مشخص کنیم که چه چیزی را می خواهیم در حافظه قرار دهیم. یک عدد صحیح کوچک، یک عدد اعشاری، یک حرف و ... .

حافظه کامپیوتر به اجزای کوچکی به نام بایت (byte) تقسیم می شود. در C++ هم کوچکترین واحد حافظه که می تواند مدیریت شود بایت است. یک بایت می تواند حاوی یک حرف و یا یک عدد کوچک باشد (عددی بین ۰ تا ۲۵۵) . علاوه بر این کامپیوتر می تواند با داده هایی که از چند بایت تشکیل شده اند نظیر اعداد اعشاری و اعداد بسیار بزرگ کار کند.

در جدول زیر شما خلاصه ای از انواع داده ای پایه و محدوده مقادیری که می توانند بگیرند را مشاهده می کنید:

نام	شرح	اندازه (byte)	محدوده
char	کاراکتر یا عدد صحیح کوچک	۱	signed: -۱۲۸ to ۱۲۷ unsigned: ۰ to ۲۵۵
short int (short)	عدد صحیح کوچک	۲	signed: -۳۲۷۶۸ to ۳۲۷۶۷ unsigned: ۰ to ٦٥٥٣٥
int	عدد صحیح	۴	signed: -۲۱٤۷٤۸۳٦٤۸ to ۲۱٤۷٤۸۳٦٤۷ unsigned: ۰ to ٤۲۹٤۹٦۷۲۹٥
long int (long)	عدد صحیح بزرگ	۴	signed: -۲۱٤۷٤۸۳٦٤۸ to ۲۱٤۷٤۸۳٦٤۷ unsigned: ۰ to ٤۲۹٤۹٦۷۲۹٥
bool	مقدار منطقی (true یا false)	۱	true or false
Float	اعداد اعشاری (ممیز شناور)	۴	۳,٤e +/- ۳۸ (۷ digits)
double	عدد اعشاری با دقت مضاعف	۸	۱,۷e +/- ۳۰۸ (۱٥ digits)
long double	عدد بزرگ اعشاری با دقت مضاعف	۸	۱,۷e +/- ۳۰۸ (۱٥ digits)

✓ اندازه متغیرها بسته به نوع کامپایلر و یا معماری کامپیوتر ممکن است متفاوت باشد.

اعلان یک متغیر

برای استفاده از متغیر ابتدا ما باید آن را اعلان یا تعریف کنیم تا مشخص شود نام متغیر چیست و از چه نوع داده ای می خواهیم استفاده کنیم. نحوه تعریف متغیر به این صورت است که ابتدا نوع و به دنبال آن نام متغیر را می نویسیم. به عنوان مثال:

```
int a;
float mynumber;
```

در خط اول یک متغیر از نوع int با نام a و در خط دوم یک متغیر از نوع float با نام mynumber تعریف شده است. پس از اعلان متغیرها می توان از آنها در حوزه ای که تعریف شده اند استفاده کرد.

اگر بخواهید چند متغیر از یک نوع را تعریف کنید می توانید در یک جمله و با جدا کردن نام متغیرها توسط کاما (,) این کار را انجام دهید. مثال:

```
int a, b, c;
```

این خط سه متغیر از نوع int را تعریف می کند. جمله بالا دقیقاً مانند جمله پایین عمل می کند:

```
int a;
int b;
int c;
```

انواع داده صحیح همانند char، short، long و int بسته به نوع محدوده ای که نیاز داریم می تواند علامت دار (signed) یا بدون علامت (unsigned) باشد. انواع علامت دار می توانند حاوی اعداد منفی و مثبت باشند در صورتی که اعداد بدون علامت تنها می توانند شامل اعداد مثبت باشند. این کار توسط قرار گرفتن کلمه signed و یا unsigned قبل از عبارت نوع داده مشخص می شود. مثال:

```
unsigned short int NumberOfSisters;
signed int MyAccountBalance;
```

اگر ما از این کلمات استفاده نکنیم کامپایلر به طور پیش فرض نوع داده ای را signed (علامت دار) در نظر می گیرد. به این ترتیب به جای خط دوم کد بالا می توانیم بنویسیم:

```
int MyAccountBalance;
```

هر دو یک معنی را می دهند. البته به غیر از نوع char که اگر بدون signed یا unsigned به کار رود مقادیر ذخیره شده در آن به عنوان کاراکتر در نظر گرفته می شوند. در صورتی که بخواهیم مقادیر عددی در این نوع داده ذخیره کنیم حتما باید از کلمات signed یا unsigned در هنگام تعریف متغیر استفاده کنیم.

کلمات short و long می توانند به تنهایی به عنوان نوع داده در نظر گرفته شوند. کامپایلر آنها را به طور پیش فرض short int و long int در نظر می گیرد. به عنوان مثال دو عبارت:

```
short Year;
short int Year;
```

با هم تفاوتی ندارند.

کلمات signed و unsigned می توانند به تنهایی به جای نوع داده قرار گیرند که به ترتیب signed int و unsigned int در نظر گرفته می شوند. دو عبارت زیر با یکدیگر برابر هستند:

```
unsigned NextYear;
unsigned int NextYear;
```

متغیرها در هر جایی از برنامه می توانند تعریف شوند و فقط زمانی می توان از یک متغیر استفاده کرد که آن را قبلاً تعریف کرده باشیم.

برنامه زیر نمونه ای ازاستفاده از متغیرها می باشد:

```
// operating with variables

#include <iostream>
using namespace std;

int main ()
{
  // declaring variables:
  int a, b;
  int result;

  // process:
  a = ٥;
  b = ٢;
  a = a + ١;
  result = a - b;

  // print out the result:
  cout << result;

  // terminate the program:
  return ٠;
}
```

نتیجه:

```
٤
```

حوزه یک متغیر

حوزه یک متغیر به محدوده ای در برنامه گفته می شود که آن متغیر در آن محدوده تعریف شده است. یک متغیر می تواند دارای حوزه سراسری (global) باشد و یا دارای حوزه محلی (local) باشد. یک متغیر سراسری در جایی درون بدنه اصلی برنامه و خارج از توابع تعریف می شود در حالی که متغیر محلی درون یک تابع یا یک بلوک تعریف می شود.

```
#include <iostream.h>

int Integer;
char aCharacter;
char string [20];
unsigned int NumberOfSons;

main ()
{
  unsigned short Age;
  float ANumber, AnotherOne;

  cout << "Enter your age:"
  cin >> Age;
  ...
}
```

متغیرهای سراسری

متغیرهای محلی

دستورالعمل ها

متغیرهای سراسری در هر جایی از برنامه می توانند مورد استفاده قرار گیرند. اما حوزه تعریف یک متغیر محلی محدود به ناحیه ای است که توسط { و } بسته و در آن ناحیه تعریف شده باشند. به عنوان مثال اگر یک متغیر در ابتدای یک تابع (مثل تابع main) تعریف شود، از آنجا تا { که تابع خاتمه می یابد متغیر تعریف شده است. اگر در یک تابع یک متغیر تعریف شود نمی توان از همان متغیر در یک تابع دیگر استفاده استفاده کرد.

مقداردهی اولیه متغیر

زمانی که یک متغیر را تعریف می کنیم مقدار آن متغیر نامشخص است. اما ممکن است شما بخواهید در همان زمان تعریف یک مقدار مشخص به آن بدهید. برای این کار شما باید متغیر را مقداردهی اولیه (initialize) کنید. در C++ دو راه برای این کار وجود دارد:

راه اول که در c هم به همین نحو است این است که بعد از نام متغیر علامت مساوی و سپس مقدار اولیه را می نویسیم:

```
type identifier = initial_value ;
```

به عنوان مثال برای تعریف یک متغیر به نام a و اختصاص مقدار اولیه صفر به آن به صورت زیر اقدام می نماییم.

```
int a = ۰;
```

راه دیگر مقدار دهی اولیه با کمک سازنده (constructor) می باشد به این صورت که بعد از نام متغیر مقدار اولیه را بین پرانتز می نویسیم.

```
type identifier (initial_value) ;
```

مثال:

```
int a (۰);
```

هر دو راه مقدار دهی اولیه یکسان و معتبر می باشد:

```
// initialization of variables
```

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

int main ()
{
  int a=٥;                // initial value = ٥
  int b(٢);               // initial value = ٢
  int result;             // initial value undetermined

  a = a + ٣;
  result = a - b;
  cout << result;

  return ٠;
}
```

نتیجه:

```
٦
```

معرفی رشته ها

متغیرهای غیر عددی که رشته ای از کاراکترها را در خود ذخیره می کنند رشته (string) نامیده می شوند. کتابخانه زبان C++ توسط کلاس string از این رشته ها پشتیبانی می کند. String یک نوع داده ای بنیادی نیست اما در اکثر کاربردهای ساده همانند این انواع داده ای عمل می کند.

اولین تفاوت این نوع با نوع داده ای بنیادی این است که برای استفاده از این نوع می بایست سرفایل (string.h (header file به برنامه اضافه شود.

```
// my first string
#include <iostream>
#include <string>
using namespace std;

int main ()
{
  string mystring = "This is a string";
  cout << mystring;
  return ٠;
}
```

نتیجه:

```
This is a string
```

همانطور که در مثال بالا می بینید نوع string نیز همانند دیگر انواع عددی تعریف و مقداردهی اولیه می شود. برای string ها هر دو نوع مقداردهی مجاز است:

```
string mystring = "This is a string";
string mystring ("This is a string");
```

مثالی دیگر از استفاده از رشته ها:

```
// my first string
```

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

int main ()
{
  string mystring;
  mystring = "This is the initial string content";
  cout << mystring << endl;
  mystring = "This is a different string content";
  cout << mystring << endl;
  return ٠;
}
```

نتیجه:

```
This is the initial string content
This is a different string content
```